Calcul analytique

Joris van Der Hoeven

Calcul analytique*

Cours pour les Journées Nationales du Calcul Formel

par Joris van der Hoeven

CNRS, LIX
École polytechnique
91128 Palaiseau Cedex
France

Email: vdhoeven@lix.polytechnique.fr
Toile: http://www.texmacs.org/joris/main/joris.html

29 novembre 2011

*. Ce travail a été subventionné par le projet ANR-09-JCJC-0098-01 MaGiX, ainsi que par la Région Ile-de-France (projet Digiteo 2009-36HD).

Table of Contents

Préface

Ce cours est une adaptation des « slides » que j'avais présentés à mon cours sur le calcul analytique aux Journées Nationales de Calcul Formel, en novembre 2011 à Luminy. J'ai rajouté un peu de matériel supplémentaire et quelques explications plus détaillées par rapport à la présentation originale, sans toutefois avoir cherché la perfection d'un cours sur papier habituel.

Pourquoi un cours sur le calcul analytique ? Traditionnellement, on choisit le calcul formel pour la beauté du calcul symbolique ! Or, les techniques du calcul formel admettent de nombreuses applications en analyse, et le calcul numérique peut parfois permettre une résolution plus efficace ou directe de problèmes en calcul formel. Toutefois, quand j'ai commencé à m'intéresser à des problèmes plus analytiques, il me manquait la culture nécessaire en analyse calculable et en arithmétique d'intervalles. Ce cours cherche à exposer les techniques les plus utiles de ces domaines, tout en gardant le lien avec le calcul formel en filigrane.

Le document inclut un certain nombre de sessions du système Mathemagix. Il s'agit de la version interprétée `mmx-light` du langage de novembre 2011. D'ici quelques années, quand l'interpréteur sera remplacé par un compilateur, il pourra être nécessaire de typer plus précisément les déclarations afin de faire fonctionner les exemples.

Remerciement. Merci à Jérémy Berthomieu pour ses commentaires sur le premier jet du manuscrit.

1

Motivation

1.1 Vers un système de calcul formel/analytique

Le projet MATHEMAGIX vise la conception et le développement d'un nouveau système de calcul formel *et* analytique [89]. À l'heure actuelle, deux types de systèmes de calcul mathématique ont connu un grand succès. De manière historique, on trouve les systèmes de calcul numérique, comme MATLAB, OCTAVE ou SCILAB. Ces logiciels permettent la résolution approchée de problèmes analytiques, comme l'intégration d'équations différentielles. D'autre part, on dispose des systèmes de calcul formel, comme MATHEMATICA, MAPLE, MAXIMA, AXIOM, SINGULAR et plus récemment SAGE. Ces logiciels visent la manipulation exacte d'objets mathématiques, en calculant directement sur des formules générales plutôt que sur des instances numériques particulières.

L'approche du calcul formel est particulièrement bien adaptée pour des objets de nature algébrique ou combinatoire — algèbre linéaire, élimination polynomiale, énumération combinatoire, théorie de Galois, etc. En revanche, les systèmes numériques classiques sont beaucoup plus efficaces pour des problèmes plus analytiques, comme l'intégration d'équations différentielles. Malheureusement, les systèmes numériques actuels opèrent principalement en précision fixe et incorporent peu d'outils pour certifier les résultats. La session suivante en MAPLE illustre bien ce problème :

```
maple> 1.000000000000000000000001 - 1;
                                    0.
```

Il est donc tentant de transposer l'esprit de calcul exact et garanti du calcul formel à l'analyse. Or le calcul formel et le calcul numérique sont encore dans une grande mesure des mondes distincts. D'une part, il y a peu d'intersections quant aux algorithmes de base utilisés (algèbre linéaire, polynômes, séries, etc.). D'autre part, il n'existe pas d'environnement de programmation convivial qui soit adapté aux deux mondes. En effet, les interactions actuelles se limitent surtout à des interfaces de type « boîte noire ».

Le calcul analytique se situe à la confluence de plusieurs domaines. L'analyse calculable, qui sera discutée dans le chapitre 2, propose une fondation réaliste au regard des architectures existantes d'ordinateurs. Un deuxième point théorique important concerne le développement d'algorithmes numériques certifiés, poursuivant des travaux classiques sur l'arithmétique des intervalles. Dans le chapitre 3, nous exposerons une variante de cette théorie : l'arithmétique de boules.

Évidemment, dans un système de calcul analytique, il est impératif de pouvoir approcher des nombres réels aussi précisément que nécessaire, donc de travailler en précision multiple. Ceci conduit à une pénalité énorme en efficacité par rapport à l'analyse numérique classique, et il convient de réduire cette pénalité autant que possible. Dans le chapitre 4, nous rappelons quelques résultats classiques en théorie de complexité. Dans les derniers chapitres 5, 6 et 7, nous combinons les différentes techniques en les appliquant à l'intégration de systèmes dynamiques et à la résolution polynomiale.

1.2 Exemples

1.2.1 Intégration numérique certifiée

Pendule

$$\ddot{y} = -\alpha \sin y,$$

Pour $\alpha = 1$, $y(0) = 0$, $\dot{y}(0) = 1$, calculer $\tilde{y} \in \mathbb{Q}$ (ou $\tilde{y} \in \mathbb{D} = \mathbb{Z}\, 2^{\mathbb{Z}}$) avec

$$|\tilde{y} - y(2011)| < 2^{-2011}.$$

Systèmes dynamiques plus généraux

Soit $P = (P_1, ..., P_d) \in \mathbb{Q}[Y]^d = \mathbb{Q}[Y_1, ..., Y_d]^d$ et considérons

$$\dot{Y} \;=\; P(Y).$$

Pour $Y(0) \in \mathbb{Q}^d$, $t \in \mathbb{Q}^>$ et $\varepsilon \in \mathbb{Q}^>$, calculer (si possible) $\tilde{Y} \in \mathbb{Q}^d$ avec

$$\left\| \tilde{Y} - Y(t) \right\| < \varepsilon.$$

Fonctions spéciales

- Fonctions élémentaires $\exp, \log, \cos, \sin, \tan$, etc.
- Fonctions holonomes : $L(f) = L_r f^{(r)} + \cdots + L_0 f = 0$, $L \in \mathbb{Q}[z][\partial]$.
- Autres : $\wp$, fonctions thêta, fonctions de Matthieu, etc.

Remarque 1.1. Il y a de nombreux autres exemples de fonctions spéciales intéressantes en théorie de nombres. Certaines de ces fonctions sont différentiellement algébriques comme ci-dessus. D'autres, comme Γ ou ζ se calculent par transformées intégrales. Voir aussi le cours de Karim Belabas pour plus de renseignements et l'utilité du calcul analytique dans ce domaine.

1.2.2 Exemple : résolution polynomiale

Étant donné un système zéro-dimensionnel

$$\begin{cases} P_1(z_1, ..., z_n) = 0 \\ \vdots \\ P_n(z_1, ..., z_n) = 0 \end{cases}$$

avec $P_1, ..., P_n \in \mathbb{Q}[z_1, ..., z_n]$, trouver les zéros "dans $\mathbb{C}^n$". Plus précisément et ici encore, il faudrait pouvoir approcher les zéros avec autant de précision que souhaitée.

Variante : prendre $P_1, ..., P_n \in \mathbb{E}[z_1, ..., z_n]$, où $\mathbb{E}$ désigne l'ensemble des "constantes exp-log", construits à partir de $\mathbb{Q}$ en utilisant les opérations $+, -, \times, \exp$ et $\log$.

Variante : chercher les racines dans $\mathbb{R}^n$.

1.2.3 Exemple : suivi de chemin

$$H(z,t) = 0 \qquad \begin{cases} H_1(z_1, ..., z_n, t) = 0 \\ \vdots \\ H_n(z_1, ..., z_n, t) = 0 \end{cases},$$

avec $H(z,t) \in \mathbb{Q}[z,t]^n$ zéro-dimensionnel en z pour $t \in \mathbb{C}\backslash\Sigma$ et Σ fini.

Étant donné (z_0, t_0) avec $H(z_0, t_0)$ et un chemin $t_0 \rightsquigarrow t_1$ qui évite Σ, calculer le chemin $z_0 \rightsquigarrow z_1$ avec $H(z_\lambda, t_\lambda) = 0$ pour tout $\lambda \in [0,1]$.

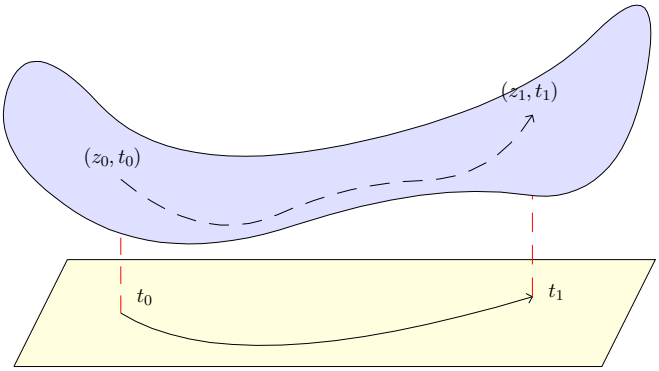


Fig. 1.1. Illustration d'un suivi de chemin.

1.2.4 Exemple : nombres de Bell

$$B(z) \;=\; \sum_{n=0}^{\infty} \frac{B_n}{n!}\, z^n = \mathrm{e}^{\mathrm{e}^z - 1}.$$

Problème 1 : calculer B_{10000} avec une erreur relative de 10^{-10}.

Problème 2 : trouver *automatiquement* le développement asymptotique

$$\log\left(\frac{B_n}{n!}\right) \;=\; n\left(-\log\log n + \frac{1}{\log n} + \frac{\log\log n}{\log n} + \mathcal{O}\left(\frac{1}{\log^2 n}\right)\right).$$

Problème 3 : pour $n \geqslant 10^{10}$, trouver une constante explicite pour le $\mathcal{O}\left(\frac{1}{\log^2 n}\right)$.

1.3 Utilité du calcul analytique pour le calcul formel

1.3.1 Résolution de systèmes polynomiaux par homotopie

La technique incontestablement la plus importante en calcul analytique est la déformation. Une application astucieuse est la résolution de systèmes polynomiaux par homotopie. Supposons par exemple que l'on veuille résoudre un système

$$P(x,y) \quad \begin{cases} x^2+2\,x\,y-y^2-3\,x+5\,y+8=0 \\ 3\,x^2-x\,y+y^2+8\,x-2\,y+7=0 \end{cases}$$

Ce système est suffisamment « générique » pour que toutes ses solutions soient simples et pour qu'il y ait exactement $4=2\times 2=\deg P_1\times\deg P_2$ solutions (comme prédit par la borne de Bézout). En particulier, le système « ressemble » (en degré, nombre et nature des solutions) au système beaucoup plus simple

$$\text{Fastoche}(x,y) \quad \begin{cases} x^2-1=0 \\ y^2-1=0 \end{cases}$$

On considère maintenant l'homotopie

$$H(x,y,t) \quad \begin{cases} (x^2-1)\,t+(x^2+2\,x\,y-y^2-3\,x+5\,y+8)\,(1-t)=0 \\ (y^2-1)\,t+(3\,x^2-x\,y+y^2+8\,x-2\,y+7)\,(1-t)=0 \end{cases}$$

qui donne le système Fastoche en $t=1$ et le système du départ P en $t=0$. Partant des solutions $x=\pm 1$, $y=\pm 1$ en $t=1$, on obtient donc les solutions de P par déformation, en suivant les solutions quand on fait bouger t de 1 vers 0 (voir la section 1.2.3 et le chapitre 7).

1.3.2 Groupe de Galois d'une fonction algébrique

La technique de suivi de chemin est aussi utile pour le calcul du groupe de Galois d'un polynôme $P\in\mathbb{Q}[t,z]$, vu comme polynôme en z sur $\mathbb{Q}(t)$. On procède comme suit :

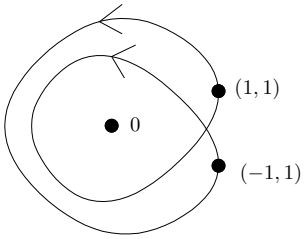
- Soit Σ l'ensemble des racines de $\operatorname{disc}_z P$.
- Soient $z_{0,1},\ldots,z_{0,d}$ les racines de P en $t_0\in\mathbb{C}\backslash\Sigma$.

Pour tout $\sigma\in\Sigma$:

- Construire un chemin t_λ^σ avec $t_0^\sigma=t_1^\sigma=t_0$ tournant autour de σ.
- Remonter en un chemin z_λ^σ avec $P(z_\lambda^\sigma,t_\lambda^\sigma)=0$ pour tout $\lambda\in[0,1]$.
- Calculer la permutation π^σ de $\{1,\ldots,d\}$ avec $z_{1,i}^\sigma=z_{0,\pi^\sigma(i)}^\sigma$.

On montre que les π^σ génèrent $\operatorname{Gal}_{\mathbb{Q}(t)}P$.

Exemple 1.2. $P = z^2 - t$.
Prendre $t_\lambda = \mathrm{e}^{2\pi\mathrm{i}\lambda}$, $z_{0,1} = 1$, $z_{0,2} = -1$.
π^0: $\begin{smallmatrix} 1 & \mapsto & 2 \\ 2 & \mapsto & 1 \end{smallmatrix}$, $\mathrm{Gal}_{\mathbb{Q}(t)} P = \mathfrak{S}_2$.



1.3.3 Groupe de Galois différentiel

Monodromie d'un opérateur différentiel

L'algorithme pour calculer le groupe de Galois d'une fonction algébrique s'adapte au cas des opérateurs différentiels $L \in \mathbb{K}(z)[\partial]$. Considérons par exemple

$$L \;=\; z\,\partial^2 + \partial$$

Base de solutions pour $L(f) = z\,f'' + f' = 0$ en $z = 1$:

$$F(z) \;=\; \begin{pmatrix} 1 \\ \log z \end{pmatrix}$$

Prolongement analytique de la solution F autour de 0:

$$F(z\,\mathrm{e}^{2\pi\mathrm{i}}) \;=\; \begin{pmatrix} 1 \\ \log z + 2\,\pi\,\mathrm{i} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2\,\pi\,\mathrm{i} & 1 \end{pmatrix} F(z).$$

La matrice de monodromie $\left(\begin{smallmatrix} 1 & 0 \\ 2\,\pi\,\mathrm{i} & 1 \end{smallmatrix}\right)$ joue un rôle analogue à la permutation π^0 de l'exemple 1.2.

Groupe de Galois différentiel

Soit $L \in \mathbb{Q}(z)[\partial]$ un opérateur différentiel linéaire qui n'admet que des singularités régulières. Alors son groupe de Galois différentiel $\mathrm{Gal}\, L$ est généré par ces matrices de monodromie en tant que groupe de algébrique fermé ; c'est le théorème de densité de Schlesinger [64, 65]. Dans le cas général, il faut rajouter les matrices de Stokes et les matrices exponentielles ; c'est le théorème de Martinet-Ramis [52].

Exemple 1.3. Dans l'exemple ci-dessus, le plus petit groupe algébrique fermé qui contient la matrice $\left(\begin{smallmatrix} 1 & 0 \\ 2\,\pi\,\mathrm{i} & 1 \end{smallmatrix}\right)$ est constitué de toutes les matrices du type $\left(\begin{smallmatrix} 1 & 0 \\ \alpha & 1 \end{smallmatrix}\right)$ avec $\alpha \in \mathbb{Q}^{\mathrm{alg}}$.

Toutes ces matrices de monodromie peuvent se calculer de façon certifiée et en temps presque linéaire [18, 77, 78]. Ceci reste vrai pour les matrices exponentielles et les matrices de Stokes [82], grace à la théorie d'accéléro-sommation d'Écalle [25, 26, 27, 12]. Pour une précision suffisamment grande de calcul, ceci conduit a un algorithme pour calculer le groupe de Galois différentiel [81]. En particulier, on obtient un algorithme de factorisation de L.

Remarque 1.4. L'algorithme de factorisation marche *grosso modo* comme ceci. Soit F un système fondamental de solutions en un point non singulier z_0 et supposons que les matrices $M_1, ..., M_k$ générant $\mathrm{Gal}\, L$ ont été calculées par rapport à z_0. L'opérateur L se factorise si et seulement si Vect F admet un sous espace vectoriel stable non trivial sous l'action de $\mathrm{Gal}\, L$, c'est à dire sous l'action de l'*algèbre* générée par $M_1, ..., M_k$.

Pour une précision de calcul donnée, si un tel espace H stable existe, il se calcule donc par algèbre linéaire. Soit $h_1, ..., h_l$ une base de H. Alors l'opérateur $K = \mathrm{ppcm}(\partial - h_1'/h_1, ..., \partial - h_l'/h_l)$ divise L. Les coefficients de K sont présentés comme des séries à coefficients dans $\mathbb{C}$, mais sont en réalité dans $\mathbb{Q}(z)$. On obtient les coefficients en tant que fractions rationnelles dans $\mathbb{Q}(z)$ par Padé-Hermite et développements en fractions continues. On montre que pour une précision de calcul suffisante, on obtient ou bien un K qui divise effectivement L dans $\mathbb{Q}(z)[\partial]$, ou bien un certificat numérique qu'il n'existe pas de sous-espace vectoriel non trivial H.

1.4 Utilité du calcul formel pour le calcul analytique

Prétraitement symbolique → meilleure efficacité

$$\mathrm{e}^{2^{-20112011}} - 1 \;\approx\; 1.0000000000 \cdot 2^{-20112011}$$

Plus généralement : asymptotique formelle → évaluation rapide de fonctions

De façon similaire, la résolution de systèmes algébriques surdéterminés peut bénéficier d'un prétraitement pour calculer une base de Gröbner ; ensuite on peut par exemple chercher les solutions numériques par méthode d'homotopie.

Calcul de schémas numériques rapides

- Runge-Kutta à des ordres élevés
- Bonnes bases de fonctions pour éléments finis
- Bonnes bases d'ondelettes

1.5 Outils communs pour calculs formel et analytique

Arithmétique rapide

- Calcul rapide en multi-précision
- Algorithmes denses rapides sur les polynômes, les matrices et les séries
- Traitement efficace de structures creuses et en évaluation

Besoin d'un bon langage

- Mathématiquement expressif
- Sémantiquement propre
- Interface conviviale
- Compilé (au moins pour le calcul analytique)

2

Analyse calculable

2.1 Nombres réels calculables

2.1.1 Définition

Comme l'ensemble $\mathbb{R}$ des nombres réels est non dénombrable, il est impossible de construire un type de données susceptible de pouvoir représenter n'importe quel nombre réel. Toutefois, au moins d'un point de vue théorique, il serait utile de pouvoir calculer directement avec des nombres réels, quitte à se restreindre à une sous-classe de $\mathbb{R}$. Le choix le plus naturel consiste à prendre la sous-classe des nombres que l'on peut *approcher* aussi précisément que nécessaire par des nombres rationnels :

Définition 2.1. *Un nombre réel $x \in \mathbb{R}$ est **calculable** s'il existe un algorithme qui prend $\varepsilon \in \mathbb{Q}^>$ en entrée et qui produit une ε-approximation $\tilde{x} \in \mathbb{Q}$ de x avec $|\tilde{x} - x| \leqslant \varepsilon$. On note $\mathbb{R}^{\mathrm{cal}}$ l'ensemble des nombres réels calculables.*

Ici, il est important d'intégrer le fait qu'un nombre est en réalité un *programme* ou encore la *promesse* de pouvoir trouver une approximation aussi fine que souhaitée. Bien sûr, la définition admet de nombreuses variantes, en jouant sur la façon d'approcher x. Par exemple :

- On peut remplacer $\mathbb{Q}$ par l'ensemble des nombres flottants
$$\mathbb{D} = \mathbb{Z}\,2^{\mathbb{Z}} = \{m\,2^e : m, e \in \mathbb{Z}\},$$
en autorisant une précision arbitraire pour les mantisses et les exposants.
- $\exists$ algorithme qui prend n en entrée et calcule une 2^{-n}-approximation.
- $\exists$ algorithme pour calculer $n_x \in \mathbb{Z}$, $x_n \in \{-1, 0, 1\}$ avec $x = \sum_{n \geqslant n_x} x_n\,2^{-n}$
- $\exists$ suites calculables $x_0^{\mathrm{lo}} \leqslant x_1^{\mathrm{lo}} \leqslant \cdots \leqslant x$, $x_0^{\mathrm{hi}} \geqslant x_1^{\mathrm{hi}} \geqslant \cdots \geqslant x$ avec $\lim_n x_n^{\mathrm{hi}} - x_n^{\mathrm{lo}} = 0$

L'analyse calculable est le sujet qui se propose de « réécrire » l'analyse sous cet angle de la calculabilité [1, 93]. La encore, il y a des sujets voisins, comme l'analyse constructive [8].

Une autre approche [71, 72, 69, 70, 21] est de faire « comme si » l'on pouvait représenter n'importe quel nombre réel et effectuer des opérations habituelles $+, -, \times, =, <$ en temps constant. Évidemment, ce modèle ne correspond encore moins à des ordinateurs concrets que les machines de Turing. Toutefois, ce modèle simplificateur n'est pas dénué de sens si on calcule en précision fixe et il n'est pas inutile d'adopter ce point de vue par moments, en particulier pour la résolution de systèmes polynomiaux (voir les chapitres 6 et 7).

2.1.2 Nombres réels calculables dans MATHEMAGIX

```
Mmx] use "asymptotix";
Mmx] type_mode? := true;
Mmx] one: Approximator (Floating, Floating) == 1;
Mmx] e == exp one
```

2.7182818284590452 : Approximator(Floating, Floating)

```
Mmx] approximate (e, 1.0e-1000)
```

2.7182818284590452353602874713526624977572470936999595749669676277240766303\
5354759457138217852516642742746639193200305992181741359662904357290033429526\
0595630738132328627943490763233829880753195251019011573834187930702154089 1\
4993488416750924476146066808226480016847741185374234544243710753907774499 20\
6955170276183860626133138458300075204493382656029760673711320070932870912 74\
4374704723069697720931014169283681902551510865746377211125238978442505695 36\
9677078544996996794686445490598793163688923009879312773617821542499922957 63\
5148220826989519366803318252886939849646510582093923982948879332036250944 31\
1730123819706841614039701983767932068328237646480429531180232878250981945 58\
1530175671736133206981125099618188159304169035159888851934580727386673858 94\
2287922849989208680582574927961048419844436346324496848756023362482704197 86\
2320900216099023530436994184914631409343173814364054625315209618369088870 70\
1676839642437814059271456354906130310720851038375051011574770417189861068 73\
9696552126715468895703503540212340784 8 : Floating

2.2 Théorèmes classiques de non calculabilité

2.2.1 Théorème de Turing

Le premier théorème de non calculabilité [75] parait naturel aujourd'hui. Or c'est ce théorème qui est à l'origine des machines de Turing et des premiers résultats de non calculabilité !

Théorème 2.2. *Il n'existe pas de test de nullité pour* $\mathbb{R}^{\mathrm{cal}}$.

Asymmétrie. Soit $x \in \mathbb{R}^{\mathrm{cal}}$.

- Si $x = 0$, alors on ne peut pas forcément le démontrer en temps fini.
- Si $x \neq 0$, alors on peut le certifier en temps fini.

Restriction à des sous-classes.

- Il existe un test de nullité pour $\mathbb{Q}^{\mathrm{clr}}$ (nombres algébriques réels).
- Il existe un test de nullité pour $\mathbb{E}$, modulo la conjecture de Schanuel.
- **Problème ouvert difficile :** test de nullité pour les constantes holonomes.

2.2.2 Théorème de Grzegorczyk

Le deuxième théorème [36, 37, 38] de non calculabilité est un peu plus surprenant, du moins pour moi-même : ce n'est pas la peine de chercher à calculer des fonctions discontinues, puisque c'est impossible !

Théorème 2.3. *Toute fonction calculable $f: \mathbb{R}^{\mathrm{cal}} \to \mathbb{R}^{\mathrm{cal}}$ est continue.*

Démonstration. Considérons une machine de Turing M pour calculer f.

Soit $x = \sum_{n \geqslant n_x} x_n 2^{-n} \in \mathbb{R}^{\mathrm{cal}}$.

Sur l'entrée (n_x, x_n), M retourne (n_y, y_n) avec $y = f(x) = \sum_{n \geqslant n_y} y_n 2^{-n}$.

Soit $\varepsilon > 0$ et $k \in \mathbb{Z}$ avec $2^{-k} < \varepsilon$, considérons le calcul **fini** de $y_{n_y}, ..., y_k$.

Soit l maximal, tel que x_l intervient dans ce calcul.

Alors $|f(x') - f(x)| < \varepsilon$ pour tout $x' \in \mathbb{R}^{\mathrm{cal}}$ avec $|x' - x| < 2^{-l}$. □

2.2.3 Théorème de Denef et Lipschitz

Le troisième théorème [22] de non calculabilité est surprenant dans ce sens que les données ne sont pas des nombres ou fonctions calculables, mais plutôt des simples vecteurs et polynômes à coefficients dans $\mathbb{Q}$. Le théorème pose clairement des limites à ce que l'on pourra calculer dans la théorie des systèmes dynamiques.

Théorème 2.4. *Considérons un système dynamique avec conditions initiales*

$$\begin{aligned} Y' &= P(Y) \\ Y(0) &\in \mathbb{Q}^d \end{aligned}$$

où $Y = (Y_1, ..., Y_d)$ et $P = (P_1, ..., P_d) \in \mathbb{Q}[Y]^d$. D'après Cauchy, ce système admet une unique solution $Y \in \mathbb{Q}[[z]]^d$. Cette solution est convergente en z.

Il n'existe pas d'algorithme pour calculer le rayon de convergence de Y_1.

2.2.4 Questionnaire

Comme exercice, le lecteur pourra essayer de déterminer lesquels des problèmes suivants sont calculables :

- Racines d'un polynôme unitaire à coefficients dans $\mathbb{C}^{\mathrm{cal}} = \mathbb{R}^{\mathrm{cal}}[\mathrm{i}]$
- Racines réelles d'un polynôme unitaire à coefficients dans $\mathbb{R}^{\mathrm{cal}}$
- Racines complexes d'un système polynomial zéro-dimensionnel sur $\mathbb{C}^{\mathrm{cal}}$
- Vecteur propres d'une matrice à coefficients dans $\mathbb{C}^{\mathrm{cal}}$
- Calcul d'une primitive d'une fonction calculable sur $\mathbb{R}^{\mathrm{cal}}$
- Calcul de la dérivée d'une fonction $\mathcal{C}^\infty$ calculable sur $\mathbb{R}^{\mathrm{cal}}$

- Calcul de la dérivée d'une fonction analytique calculable sur $\mathbb{C}^{\mathrm{cal}}$
- Calcul d'une racine de $P \in \mathbb{R}^{\mathrm{cal}}[X]$ avec $P(0)\,P(1) < 0$

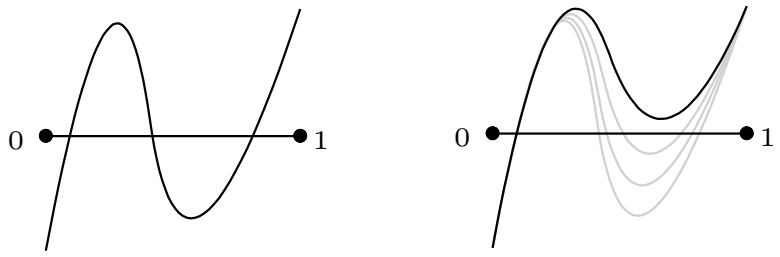


Fig. 2.1. Démonstration qu'il n'existe pas de fonction calculable

$$\text{racine}\colon \{P \in \mathbb{R}^{\mathrm{cal}}[X]\colon P(0)\,P(1) < 0\} \to [0,1]^{\mathrm{cal}}$$

avec $P(\text{racine}(P)) = 0$. En effet, dans l'image à gauche, une telle fonction devrait pouvoir nous fournir une racine de P, disons le point du milieu. Par le théorème de Grzegorczyk, si on déforme P continument, on doit pouvoir suivre cette racine. Mais dans l'image à droite, on voit que l'on peut alors faire disparaître la racine. Contradiction.

Remarque 2.5. Subtilité concernant le dernier problème : il existe une fonction qui prend en entrée une *représentation* $\check{P}$ d'un polynôme $P \in \mathbb{R}^{\mathrm{cal}}[X]$ avec $P(0)\ P(1) < 0$, et qui retourne une racine $x \in [0,1]^{\mathrm{cal}} = [0,1] \cap \mathbb{R}^{\mathrm{cal}}$ de P. Le hic réside dans le fait que cette fonction peut retourner deux racines distinctes pour deux représentations différentes du même polynôme P.

2.3 Semi calculabilité et typage

2.3.1 Autres types de calculabilité

Les théorèmes de non calculabilité de la section 2.2 sont inutilement pessimistes. Par exemple, le théorème de Turing ne rend pas compte du fait suivant : étant donné un nombre $x \in \mathbb{R}^{\mathrm{cal}}$, on peut démontrer en temps fini que $x \neq 0$, si tel est en effet le cas. Pour mieux cerner ce qui reste calculable, il est utile d'introduire les ensembles de nombres réels calculables à gauche et à droite, et qui reflètent mieux l'asymmétrie profonde de l'analyse calculable entre l'égalité et l'inégalité.

Calculabilité à gauche

$x \in \mathbb{R}^{\mathrm{calg}}$ s'il existe $\check{x}_0 \leqslant \check{x}_1 \leqslant \cdots$ calculable avec $\check{x}_n \in \mathbb{D}$ et $\lim_{n\to\infty} x_n = x$

Calculabilité à droite

$x \in \mathbb{R}^{\mathrm{cald}}$ si $-x \in \mathbb{R}^{\mathrm{calg}}$. On a $\mathbb{R}^{\mathrm{cal}} = \mathbb{R}^{\mathrm{calg}} \cap \mathbb{R}^{\mathrm{cald}}$

On peut introduire une ribambelle de notions de calculabilité en poursuivant sur cette voie. Par exemple :

Inférieurement calculable

$X \in \mathscr{P}^{\text{calinf}}(\mathbb{R}^d)$ s'il existe $\check{X}_0 \subseteq \check{X}_1 \subseteq \cdots$ calculable avec $\bigcup_{n\in\mathbb{N}} \check{X}_n = X$, où chaque $\check{X}_n$ est une réunion finie de « blocks » fermés à extrémités dans $\mathbb{D}^d$

Supérieurement calculable

$X \in \mathscr{P}^{\text{calsup}}(\mathbb{R}^d)$ si $\mathbb{R}^d \setminus X \in \mathscr{P}^{\text{calinf}}(\mathbb{R}^d)$

2.3.2 Typage

Il est utile de considérer des ensembles comme $\mathbb{R}^{\text{cal}}$, $\mathbb{R}^{\text{calg}}$, $\mathbb{R}^{\text{cald}}$, $\mathscr{P}^{\text{calinf}}(\mathbb{R}^d)$, etc. comme des types de données *concrets*, pouvant intervenir dans des implantations. On peut d'ailleurs construire plein d'autres types du même acabit, comme les booléens à droite $\{0,1\}^{\text{cald}} = \mathbb{R}^{\text{cald}} \cap \{0,1\}$ ou les « booléens modulo la conjecture de Schanuel » $\{0,1\}^{\text{Schanuel}}$.

Ces types de données plus fins permettent de transformer les énoncés quelque peu négatives de la section 2.2 en assertions plus positives :

- $=: \mathbb{R}^{\text{cal}} \times \mathbb{R}^{\text{cal}} \to \{0,1\}^{\text{cald}}$ calculable
- $f: \mathbb{R}^{\text{cal}} \to \mathbb{R}^{\text{cal}}$ calculable implique $\text{graphe}(f) \in \mathscr{P}^{\text{calsup}}(\mathbb{R}^2)$
- rayonsol: $\mathbb{Q}[Y]^d \times \mathbb{Q}^d \to \mathbb{R}^{\text{calg}}$ calculable
- racines: $\mathbb{C}^{\text{cal}}[z]_{\text{unit}} \to \mathscr{M}^{\text{cal}}(\mathbb{C})$ calculable ($\mathscr{M}$ pour multi-ensembles)
- racines: $\mathbb{R}^{\text{cal}}[z]_{\text{unit}} \to \mathscr{P}^{\text{calsup}}(\mathbb{R})$ calculable
- racines: $\mathscr{F}([0,1]^{\text{cal}}, \mathbb{R}^{\text{cal}}) \to \mathscr{P}^{\text{calsup}}(\mathbb{R})$ calculable
- $=: \mathbb{E} \times \mathbb{E} \to \{0,1\}^{\text{Schanuel}}$ calculable (égalité modulo conjecture de Schanuel)

2.3.3 Structures effectives

D'un point de vue logique, les types de données peuvent être vu comme des « structures effectives » sur un ensemble E. Chaque élément de E admet alors une représentation dans un ensemble $\check{E}$ de représentations. Chaque représentation peut s'encoder à son tour par un entier dans $\mathbb{N}$, et donc être manipulée par des machines de Turing.

$$E \xleftarrow{\text{représentation}} \check{E} \xrightarrow{\text{encodage}} \mathbb{N}$$

Par exemple, un nombre dans $\mathbb{R}^{\text{calg}}$ pourrait être représenté par une suite calculable croissante :

$$\begin{array}{ccccc} \mathbb{R}^{\text{calg}} & \xleftarrow{\text{représentation}} & \widetilde{\mathbb{R}^{\text{calg}}} & \xrightarrow{\text{encodage}} & \mathbb{N} \\ x = \lim\limits_{n\to\infty} \check{x}_n & & (\check{x}_n) \in \mathscr{F}^{\text{cal}}(\mathbb{N}, \mathbb{D})\uparrow & & \end{array}$$

Il est à noter que, de même que l'on peut définir plusieurs structures de groupe sur un ensemble $\{a, b, c, d\}$ à quatre éléments, on peut définir plusieurs structures effectives sur $\{0,1\}$: on a bien sûr $\{0,1\} = \{0,1\}^{\mathrm{cal}}$, mais aussi $\{0,1\}^{\mathrm{calg}}$ et $\{0,1\}^{\mathrm{cald}}$. La représentation d'un élément de $\{0,1\}^{\mathrm{Schanuel}}$ pourrait être une fonction qui retourne un élément de $\{0,1\}$ si la conjecture de Schanuel est vraie.

2.3.4 Approximateurs

En analyse, les objets que l'on manipule correspondent généralement à des résultats de processus d'approximation. Il s'agit donc de structures effectives d'une nature particulière :

- $\tilde{E}$: ensemble abstrait d'approximations d'éléments dans E
- $x = \lim_{n\to\infty} \tilde{x}_n$: $x \in E$ est la limite des approximations $\tilde{x}_n$
- $\check{E} = \big\{(\check{x}_n) \in \mathscr{F}^{\mathrm{cal}}\big(\mathbb{N}, \tilde{E}\big) : (\check{x}_n) \text{ admet une limite } x \in E\big\}$

Le plus souvent, $\tilde{E}$ est constitué de parties de E (comme des intervalles ou des boules dans le cas où $E \subseteq \mathbb{R}$) et la notion de limite correspond à prendre une intersection. Par exemple :

- $\widetilde{\mathbb{R}^{\mathrm{cal}}} = \{[a,b] : a, b \in \mathbb{D}, a \leqslant b\}$
- $x = \lim_{n\to\infty} \tilde{x}_n$ si $\{x\} = \bigcap_{n\in\mathbb{N}} \tilde{x}_n$
- $\widetilde{\mathbb{R}^{\mathrm{cal}}} = \Big\{(\check{x}_n) \in \mathscr{F}^{\mathrm{cal}}\Big(\mathbb{N}, \widetilde{\mathbb{R}^{\mathrm{cal}}}\Big) : \exists x \in \mathbb{R}, \{x\} = \bigcap_{n\in\mathbb{N}} \tilde{x}_n\Big\}$

Toutefois, il faut parfois tordre un peu la notion de limite :

- $\widetilde{\mathbb{R}^{\mathrm{calg}}} = \{[a,\infty[: a \in \mathbb{D}\}$
- $x = \lim_{n\to\infty} \tilde{x}_n$ si $x = \inf \bigcap_{n\in\mathbb{N}} \tilde{x}_n$
- $\widetilde{\mathbb{R}^{\mathrm{calg}}} = \Big\{(\check{x}_n) \in \mathscr{F}^{\mathrm{cal}}\Big(\mathbb{N}, \widetilde{\mathbb{R}^{\mathrm{calg}}}\Big) : \bigcap_{n\in\mathbb{N}} \tilde{x}_n \neq \varnothing\Big\}$

L'intérêt de cette formalisation est une certaine fonctorialité. Par exemple, si on peut approcher les éléments de E et de F, il en est de même pour les éléments de $E \times F$ et les fonctions dans $\mathscr{F}(E,F)$ de E vers F :

$$\widetilde{E \times F} = \tilde{E} \times \tilde{F} \qquad \tilde{x} \subseteq E, \tilde{y} \subseteq F \Rightarrow \tilde{x} \times \tilde{y} \subseteq E \times F$$

$$\widetilde{\mathscr{F}^{\mathrm{cal}}(E,F)} = \mathscr{F}^{\mathrm{cal}}\big(\tilde{E}, \tilde{F}\big) \qquad f : E \to F \in \tilde{f} : \tilde{E} \to \tilde{F} \text{ si } x \in \tilde{x} \Rightarrow f(x) \in \tilde{f}(\tilde{x})$$

3

Arithmétique de boules

3.1 Principes

3.1.1 Encadrements par des boules

Dans la définition 2.1, nous avions choisi d'approcher les nombres réels par des rationnels. Malheureusement, de telles approximations ne sont pas des plus judicieuses si nous voulons borner de façon automatique les erreurs intervenant dans un calcul complexe.

Pour cette raison (voir aussi la section 2.3.4), il est plus commode d'approcher un nombre réel donné $x \in \mathbb{R}$ non pas par d'autres nombres $\tilde{x}$ avec $|\tilde{x} - x|$ « petit », mais plutôt par des boules fermées $x^\bullet \ni x$ de la forme

$$x^\bullet = \mathcal{B}(c, r) = c + \mathcal{B}(r) \;=\; \{x \in \mathbb{R} \colon |x - c| \leqslant r\},$$

avec $c \in \mathbb{R}$ et $r \in \mathbb{R}^{\geqslant}$. Ainsi, lorsque $x^\bullet$ est le résultat d'un calcul par arithmétique de boules, on est *sûr* que le « vrai résultat » x est dans $x^\bullet$. On peut donc interpréter une boule $x^\bullet$ comme un « nombre x que l'on ne connait pas, mais pour lequel on est *sûr* que $x \in x^\bullet$ ».

Remarque 3.1. Ce principe de calcul n'est pas tout à fait nouveau d'un point de vue historique. En effet, on peut observer que les notations $\mathcal{O}$ et o de Landau s'interprètent de la même manière. Par exemple, dans $f(z) = 1 + z + \frac{1}{2} z^2 + \mathcal{O}(z^3)$, le terme $\mathcal{O}(z^3)$ désigne « une quantité f que l'on ne connaît pas, mais pour lequel on est sûr que $\exists K, \exists \varepsilon > 0, \forall |z| < \varepsilon$, $|f(z)| \leqslant K z^3$ ».

Remarque 3.2. Au lieu d'encadrer les nombres x par des boules, on peut aussi les encadrer par des intervalles fermées $x^\vdash = [x^{\mathrm{lo}}, x^{\mathrm{hi}}]$. Ceci conduit à l'arithmétique d'intervalles, qui est plus classique et pour laquelle existe une littérature abondante [54, 2, 57, 40, 9, 63] ; voir la section 3.1.4 pour une comparaison. L'arithmétique de boules est parfois appelé « arithmétique d'intervalles par centres et rayons » ou « arithmétique d'intervalles circulaires ». J'ai préféré un nom plus court, d'autant plus qu'une boule complexe (par exemple) n'a rien à voir avec des intervalles.

Bien sûr, on peut prendre des boules dans des espaces métriques plus généraux que $\mathbb{R}$, en commençant par $\mathbb{C}$. En fait, il n'est même pas impératif que les rayons des boules vivent dans un espace métrique au sens classique. Plus tard, on verra par exemple l'utilité de considerer des boules dont les centres sont des matrices dans $\mathbb{R}^{n \times n}$ et dont les rayons sont également des matrices à coefficients positifs dans $(\mathbb{R}^{\geqslant})^{n \times n}$.

Dans la suite, étant donnés un ensemble C de centres et un ensemble R de rayons, on notera par $\mathcal{B}(C,R)$ l'ensemble des boules avec des centres dans C et des rayons dans $R^{\geqslant}$.

3.1.2 Opérations

Soit $f\colon \mathbb{R}^d \to \mathbb{R}$ une fonction. Une fonction $f^\bullet\colon \mathcal{B}(\mathbb{R},\mathbb{R})^d \to \mathcal{B}(\mathbb{R},\mathbb{R})$ est appelée une *extension* de $f\colon \mathbb{R}^d \to \mathbb{R}$ si

$$f(x_1^\bullet, ..., x_d^\bullet) = \{ f(x_1, ..., x_d) \colon x_1 \in x_1^\bullet, ..., x_n \in x_n^\bullet \} \ \subseteq \ f^\bullet(x_1^\bullet, ..., x_n^\bullet), \tag{3.1}$$

pour tous les $x_1^\bullet$, ..., $x_d^\bullet \in \mathcal{B}(\mathbb{R}, \mathbb{R})$. Ainsi, si $x_1 \in x_1^\bullet$, ..., $x_n \in x_n^\bullet$, alors on peut être *sûr* que $f(x_1^\bullet, ..., x_d^\bullet) \subseteq f^\bullet(x_1^\bullet, ..., x_n^\bullet)$, conformément à la sémantique décrite dans la section précédente.

On peut utiliser les formules suivantes pour les opérations élémentaires :

$$\mathcal{B}(x,r) +^\bullet \mathcal{B}(y,s) \ = \ \mathcal{B}(x+y, r+s) \tag{3.2}$$
$$\mathcal{B}(x,r) -^\bullet \mathcal{B}(y,s) \ = \ \mathcal{B}(x-y, r+s) \tag{3.3}$$
$$\mathcal{B}(x,r) \times^\bullet \mathcal{B}(y,s) \ = \ \mathcal{B}(x \times y, r \times (|y|+s) + |x| \times s) \tag{3.4}$$

Il est à noter que la dernière formule est simple, mais pas nécessairement optimale : par exemple, on trouve $\mathcal{B}(1,1) \times^\bullet \mathcal{B}(1,1) = \mathcal{B}(1,3)$, alors que la boule $\mathcal{B}(2,2) \subsetneq \mathcal{B}(1,3)$ vérifie bien $x, y \in \mathcal{B}(1,1) \Rightarrow x\,y \in \mathcal{B}(2,2)$.

3.1.3 Opérations en précision limitée

Pour des calculs concrets sur machine, on ne peut prendre les centres et les rayons dans $\mathbb{R}$. Dans ce cas, on considère plutôt des centres et des rayons dans $\mathbb{D}$, voire dans l'ensemble

$$\mathbb{D}_{p,e} \ = \ \pm\{0, ..., 2^p - 1\}\, 2^{\pm\{0,...,2^e-1\}} \cup \{\pm\infty\}$$

des nombres flottants avec un mantisse de p bits et un exposant de e bits. Dans ce dernier cas, il est à noter qu'il est impératif de rajouter la possibilité de prendre des rayons infinis, afin de représenter le résultat d'un *overflow* (dépassement de précision).

Quand on calcule avec en précisions p et e fixes, le résultat exact $y = f(x_1, ..., x_d)$ d'une opération sur des nombres x_1, ..., x_d dans $\mathbb{D}_{p,e}$ n'est pas nécessairement dans $\mathbb{D}_{p,e}$ et son approximation $\tilde{y}$ par un élément dans $\mathbb{D}_{p,e}$ induit donc une erreur $|\tilde{y} - y|$ dont il faut tenir compte dans l'implantation d'une arithmétique de boules.

La norme IEEE [3, 56] pour le calcul avec des nombres flottants spécifie que l'approximation $\tilde{y}$ est obtenue en arrondissant le résultat exact y suivant un mode d'arrondi à spécifier. Nous utiliserons les notations $\uparrow$, $\downarrow$, $\updownarrow$ pour les modes d'arrondi vers le haut, vers le bas et au plus près. En notant $\mathrm{next}(a) = \min\{b \in \mathbb{D}_{p,e} \colon b > a\}$, nous notons que $\mathrm{errarr}(\tilde{y}) = \mathrm{next}(|\tilde{y}|) - |\tilde{y}|$ est une borne supérieure pour l'erreur quelque soit le mode d'arrondi choisi. Les formules (3.2), (3.3) et (3.4) peuvent donc adaptées au cas du calcul en précision limitée, en les remplaçant par

$$\begin{aligned}
\mathcal{B}(x,r) +^\bullet \mathcal{B}(y,s) \ &= \ \mathcal{B}\big(x +^\updownarrow y, r +^\uparrow s +^\uparrow \mathrm{errarr}\big(x +^\updownarrow y\big)\big) \\
\mathcal{B}(x,r) -^\bullet \mathcal{B}(y,s) \ &= \ \mathcal{B}\big(x -^\updownarrow y, r +^\uparrow s +^\uparrow \mathrm{errarr}\big(x -^\updownarrow y\big)\big) \\
\mathcal{B}(x,r) \times^\bullet \mathcal{B}(y,s) \ &= \ \mathcal{B}\big(x \times^\updownarrow y, r \times^\uparrow (|y| +^\uparrow s) +^\uparrow |x| \times^\uparrow s +^\uparrow \mathrm{errarr}\big(x \times^\updownarrow y\big)\big)
\end{aligned}$$

Il est à noter que la norme IEEE est désormais suivie par la plupart des constructeurs de micro-processeurs. MPFR fournit aussi une bonne implantation en précision arbitraire [39].

3.1.4 Arithmétique d'intervalles

On a déjà mentionné le fait que l'arithmétique de boules est classiquement considérée comme une variante de l'arithmétique d'intervalles. Pourquoi choisir l'une ou l'autre ? Voici quelques avantages et inconvénients des deux approches.

Avantages de l'arithmétique d'intervalles

- Quand les intervalles sont grands, l'arithmétique d'intervalles est généralement plus précises. Pour certaines applications, les intervalles sont grands par nature et donc plus adaptées. Par exemple, lors de la recherche de solutions d'un système d'équations dans une boîte, il est tout à fait adapté d'utiliser une méthode de découpage par dichotomie.
- Toute fonction monotone $f: \mathbb{R} \to \mathbb{R}$, disons croissante, admet une extension canonique $f^{\mapsto}: \mathcal{I}(\mathbb{D}_{p,e}) \to \mathcal{I}(\mathbb{D}_{p,e})$ en arithmétique d'intervalles : $f^{\mapsto}([x^{\mathrm{lo}}, x^{\mathrm{hi}}]) = [f^{\downarrow}(x^{\mathrm{lo}}), f^{\uparrow}(x^{\mathrm{hi}})]$. Ceci favorise la standardisation.
- En implantant la norme IEEE, les processeurs modernes privilégient *a priori* l'efficacité de l'arithmétique d'intervalles. Attention toutefois : changer le mode d'arrondi peut s'avérer très coûteux en temps, auquel cas il faut adapter les algorithmes afin de pouvoir fonctionner avec n'importe quel mode d'arrondi [92, 45].

Avantages de l'arithmétique de boules

- En précision multiple, les intervalles sont généralement petits, ce qui permet le stockage du centre en précision multiple et le rayon en précision simple. Ceci est deux fois plus efficace en temps et en espace par rapport à l'arithmétique d'intervalles qui oblige à stocker les deux extrémités en précision multiple. Si on cherche à approcher de plus en plus précisément des vrais nombres réels, il est donc plus naturel d'utiliser des boules.
- L'arithmétique de boules donne une grande souplesse quant au choix des centres et des rayons. D'une part, ceci peut être utilisé pour améliorer la qualité des estimations (voir la section 3.2.3). D'autre part, ceci peut être utilisé pour améliorer la vectorisation et plus généralement l'efficacité des algorithmes (voir la section 3.4.2).
- En mathématiques, les estimations d'erreur s'établissent généralement *via* le calcul ε-δ classique. Ce calcul correspond naturellement à l'arithmétique de boules, ce qui facilite la conception d'algorithmes certifiés. Ceci s'applique tout particulièrement à tout argument par perturbation.

3.1.5 Prédicats

Une question intéressante concerne la sémantique des prédicats comme $=$. Si on interprète les boules comme un ensemble de « valeurs possibles », alors le résultat d'un test $x^\bullet = y^\bullet$ devrait être un intervalle $b^\bullet$ de $[0, 1]$ avec

$$x \in x^\bullet \wedge y \in y^\bullet \;\Rightarrow\; (x = y) \in b^\bullet.$$

En revanche, si on interprète $x^\bullet$ et $y^\bullet$ comme des approximation d'un vrai nombre réel à une certaine précision, il est plus naturel de prendre

$$x^\bullet = y^\bullet \iff x^\bullet \cap y^\bullet \neq \varnothing.$$

En effet, si $x, y \in \mathbb{R}^{\mathrm{cal}}$ sont des nombres calculables représentés par des suites calculables décroissantes de boules $x_p^\bullet$, $y_p^\bullet$ avec $\{x\} = \bigcap_p x_p^\bullet$ et $\{y\} = \bigcap_p y_p^\bullet$, alors la suite $x_p^\bullet = y_p^\bullet$ à valeurs dans $\{0,1\}$ représente le résultat du test d'égalité $x = y$ dans $\{0,1\}^{\mathrm{calr}}$. La deuxième définition reflète donc l'asymmétrie profonde de l'égalité dans l'analyse calculable.

3.1.6 Estimations d'erreur *a priori*

L'arithmétique des boules est basée sur une analyse *a posteriori* des erreurs : on effectue une opération en arithmétique de boules en utilisant une certaine précision que l'on augmentera jusqu'au moment où le rayon du résultat est suffisamment petit. Parfois, bien que rarement, on peut préférer une estimation *a priori* de l'erreur [80].

Exemple : calcul d'une ε-approximation de $z = x + y$

Calculer des $\frac{\varepsilon}{2}$-approximations $\tilde{x}$ et $\tilde{y}$ de x et y
Retourner $\tilde{x} + \tilde{y}$

Avantage : précision de calcul adaptatif dans le case $1 + 10^{-100}$

En effet, dans l'arithmétique de boules classique, on calcule 1 et 10^{-100} en utilisant la même précision, alors que l'on peut simplement utiliser 0 comme approximation de 10^{-100} tant que $\varepsilon \geqslant 2 \times 10^{-100}$. Supposant que 10^{-100} est un nombre dont l'approximation est très couteuse en temps, une estimation *a priori* de l'erreur est donc plus efficace.

Problème : évaluation de $P_d x^d + \cdots + P_0$ par Horner en $x = 1$

Calcul d'une $\frac{\varepsilon}{2^{d-k}}$-approximation de P_k pour $k = 0, \ldots, d-1$

$\longrightarrow$ tolérance trop petite

Équilibrage des tolérances : calcul d'une $\frac{\varepsilon}{d+1}$-approximation de P_k pour tout k

Problème : nécessite des « dags » (*directed acyclic graphs*) pour tous les résultats intermédiaires

3.2 Limiter la surestimation

3.2.1 L'ennemi

Surestimation de l'erreur

Le problème principal de l'arithmétique de boules concerne la surestimation des erreurs. Par exemple, l'évaluation de la fonction $f(x) = x - x$ en $\mathcal{B}(0,1)$ donne

$$\mathcal{B}(0,1) - \mathcal{B}(0,1) \;=\; \mathcal{B}(0,2).$$

Plus généralement, ce problème intervient chaque fois qu'il y a des quantités qui s'annulent lors d'un calcul, sans que la méthode de calcul s'en rend compte.

Effet d'enveloppement

Dans certain cas, la surestimation est due à notre façon de représenter les encadrements. Par exemple, en arithmétique d'intervalles standard, on encadre les nombres complexes par des rectangles de la forme $[x^{\mathrm{lo}}, x^{\mathrm{hi}}] + [y^{\mathrm{lo}}, y^{\mathrm{hi}}]\,\mathrm{i}$. Ces rectangles ont tendance à se tourner lors de la multiplication par un nombre complexe non réel, comme dans l'exemple

$$([1-\varepsilon, 1+\varepsilon] + [1-\varepsilon, 1+\varepsilon]\,\mathrm{i})^2 \;=\; [-2\,\varepsilon, 2\,\varepsilon] + [2-2\,\varepsilon, 2+2\,\varepsilon]\,\mathrm{i}. \tag{3.5}$$

Ici l'inclusion du résultat dans un autre rectangle de la forme requise implique une perte automatique de précision. On appelle ceci l'*effet d'enveloppement*. Sur cet exemple précis, l'inconvénient disparaît lorsque l'on utilise l'arithmétique de boules, puisqu'une boule reste une boule si on la tourne. Toutefois, on verra un exemple plus complexe dans la section 5.5.2, où il faudra travailler plus pour résoudre le problème.

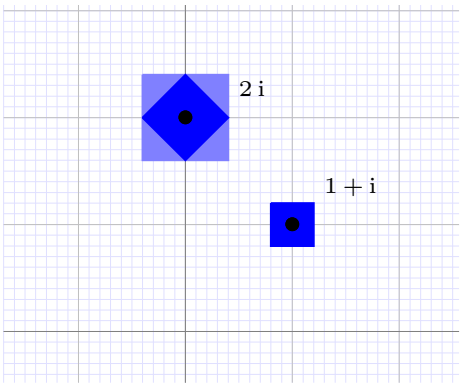


Fig. 3.1. Illustration de l'effet d'enveloppement pour l'exemple (3.5).

3.2.2 Le problème pour le calcul des puissances successives

Illustrons la perte de précision lorsque l'on calcule $(1+\mathrm{i})^k$ pour $k = 4, 8, 12, \ldots$ de façon naïve en utilisant l'arithmétique d'intervalles. Dans la sortie, on utilise la notation scientifique. Par exemple 4096.00 désigne un nombre compris entre 4095.995 et 4096.005.

```
Mmx] use "algebramix";
Mmx] puissance (z, n) ==
       if n = 1 then z else z * puissance (z, n-1);
Mmx] z == complex (interval (1.0, 1.0000000001),
                   interval (1.0, 1.0000000001))
```

$1.000000000 + 1.000000000\,\mathrm{i}$

```
Mmx] [ puissance (z, 4*n) || n in 1 to 10 ]
```

$$\begin{bmatrix} -4.00000000 \\ 16.0000000 \\ -64.000000 \\ 256.0000 \\ -1024.000 \\ 4096.00 \\ -16384.0 \\ 6.5536e4 \\ -2.62e5 \\ 1.05e6 \end{bmatrix}$$

3.2.3 Choix d'une bonne représentation par boules complexes

Comme on l'avait prédit, la perte de précision n'apparait pas lorsque l'on remplace l'arithmétique d'intervalles par l'arithmétique de boules.

```
Mmx] use "algebramix";
Mmx] puissance (z, n) ==
       if n = 1 then z else z * puissance (z, n-1);
Mmx] z == ball (complex (1.0, 1.0), 0.0000000001)
```

$$1.000000000 + 1.000000000\,\mathrm{i}$$

```
Mmx] [ puissance (z, 4*n) || n in 1 to 10 ]
```

$$\begin{bmatrix} -4.00000000 \\ 16.0000000 \\ -64.000000 \\ 256.000000 \\ -1024.00000 \\ 4096.0000 \\ -16384.0000 \\ 65536.000 \\ -262144.00 \\ 1.04857600e6 \end{bmatrix}$$

3.2.4 Minimisation de la profondeur du calcul

Une autre technique pour limiter l'effet d'enveloppement consiste à utiliser des algorithmes qui minimisent la profondeur du calcul. En effet, dans la section 3.2.2, l'effet est amplifié par le fait que l'on réinjecte k fois le résultat du calcul précédent dans l'étape d'après, perdant ainsi $\mathcal{O}(k)$ bits de précision. Si on utilise un algorithme diviser pour régner pour calculer des puissances, la perte de précision se limite à $\mathcal{O}(\log k)$ bits. À noter que ce genre d'algorithmes sont intéressants de toute façon, car ils se parallèlisent généralement mieux et ils se comportent mieux vis à vis de la mémoire cache.

```
Mmx] use "algebramix";
Mmx] puissance (z, n) ==
       if n = 1 then z
       else puissance (z, n quo 2) * puissance (z, n - (n quo 2));
Mmx] z == complex (interval (1.0, 1.0000000001), interval (1.0,
     1.0000000001))
```

$$1.000000000 + 1.000000000\,\mathrm{i}$$

```
Mmx] [ puissance (z, 4*n) || n in 1 to 10 ]
```

$$\begin{bmatrix} -4.00000000 \\ 16.0000000 \\ -64.000000 \\ 256.000000 \\ -1024.00000 \\ 4096.0000 \\ -16384.000 \\ 65536.000 \\ -262144.00 \\ 1.0485760e6 \end{bmatrix}$$

3.2.5 La méthode perturbative

Nous avons déjà souligné l'importance des méthodes perturbatives en calcul analytique. Elles interviennent notamment lorsque l'on cherche à résoudre des équations et que l'on a déjà un moyen pour résoudre l'équation de façon approchée.

Considérons par exemple l'inversion d'une matrice M, qui correspond à la résolution de l'équation $MN = 1$. Si $M = \mathcal{B}(C,R) \in \mathcal{B}(\mathbb{D},\mathbb{D})^{n\times n}$ est une matrice de boules, réécrite comme une boule de centre $C \in \mathbb{D}^{n\times n}$ et de rayon $R \in (\mathbb{D}^{\geqslant})^{n\times n}$, le calcul de M^{-1} par pivot de Gauss naïf produit une surestimation énorme. Or il existe de bons algorithmes numériques pour inverser le centre C de façon approchée, produisant une matrice U avec $CU - 1 \approx 0$. Dès lors, il suffit d'étudier de combien l'inverse de C peut bouger lorsque l'on fait varier C. Ceci conduit à l'algorithme suivant :

Algorithme (Hansen)
Entrée : $M \in \mathcal{B}(\mathbb{D},\mathbb{D})^{n\times n}$
Sortie : l'inverse $M^{-1} \in \mathcal{B}(\mathbb{D},\mathbb{D})^{n\times n}$ de M

- Écrire $M = \mathcal{B}(C,R)$, avec $C \in \mathbb{D}^{n\times n}$ et $R \in (\mathbb{D}^{>})^{n\times n}$
- Inverser numériquement $U \approx C^{-1}$
- Calculer $E = C\mathcal{B}(U,0) - \mathcal{B}(1,0)$
- Calculer $\varepsilon = \|E\|_\infty = \max_{1\leqslant i\leqslant n}(|E_{i1}| + \cdots + |E_{in}|)$. On a :

$$\|(1+E)^{-1} - 1\|_\infty = \|-E + E^2 - E^3 + \cdots\|_\infty \leqslant \frac{\varepsilon}{1-\varepsilon}$$

- Soit $\mathbb{1}$ la matrice avec entrées $\mathbb{1}_{ij} = 1$
- Retourner $U\mathcal{B}(1, \frac{\varepsilon}{1-\varepsilon}\mathbb{1})$

Remarque. De manière plus générale, la méthode perturbative appartient à la famille des algorithmes marchant par « prospection-validation ». Dans un premier temps, ces algorithmes cherchent à approcher, voire à diviner la bonne réponse. Lorsque l'on estime avoir accumulé suffisamment d'évidence pour que le résultat soit bon, on lance un nouvel algorithme pour la validation du résultat.

3.2.6 Calcul d'une forme échelon certifiée dans Mathemagix

Bien sûr, la méthode perturbative s'applique à d'autres problèmes similaires, comme le calcul de la forme échelon. Montrons d'abord le calcul quand on utilise l'algorithme naïf qui consiste à directement appliquer le pivot de Gauss sur une matrice dont les coefficients sont des intervalles :

```
Mmx] use "analyziz"
Mmx] rnd () == {
       x == uniform_deviate (0.0, 1.0);
       return interval (x - 0.0000001, x + 0.0000001);
     };
Mmx] M == [ rnd () | j in 1 to 7 || i in 1 to 7 ]
```

$$\begin{bmatrix} 0.150113 & 0.115819 & 0.197518 & 0.923428 & 0.423082 & 0.052249 & 0.612066 \\ 0.019544 & 0.567910 & 0.752122 & 0.607832 & 0.174048 & 0.942806 & 0.11206 \\ 0.368797 & 0.510038 & 0.53223 & 0.551924 & 0.825280 & 0.041178 & 0.004108 \\ 0.989109 & 0.82567 & 0.698330 & 0.418762 & 0.363986 & 0.938910 & 0.693872 \\ 0.02625 & 0.106969 & 0.631864 & 0.591872 & 0.57353 & 0.428475 & 0.806725 \\ 0.001036 & 0.709038 & 0.254791 & 0.60645 & 0.739154 & 0.535596 & 0.392901 \\ 0.383497 & 0.517670 & 0.445796 & 0.324897 & 0.51565 & 0.779884 & 0.685690 \end{bmatrix}$$

```
Mmx] row_echelon M
```

$$\begin{bmatrix} 0.150113 & 0.115819 & 0.197518 & 0.923428 & 0.423082 & 0.052249 & 0.612066 \\ 0e-6 & 0.55283 & 0.72641 & 0.48761 & 0.1190 & 0.93600 & 0.03237 \\ 0e-6 & 0e-5 & -0.24932 & -1.9156 & -0.26267 & -0.4690 & -1.5128 \\ 0e-5 & 0e-5 & 0e-4 & -0.46 & -1.7152 & 1.7778 & 0.82 \\ 0e-4 & 0e-4 & 0e-3 & 0e-2 & 12.6 & -13.75 & -8.3 \\ 0e-4 & 0e-3 & 0e-3 & 0e-2 & 0e-1 & 1.0 & 1.8 \\ 0e-4 & 0e-3 & 0e-3 & 0e-2 & 0e-1 & 0e-1 & -0.3 \end{bmatrix}$$

On observe bien une déperdition de la précision au cours de l'algorithme. Lorsque les coefficients de la matrice sont remplacés par des boules, Mathemagix utilise la méthode perturbative :

```
Mmx] use "analyziz"
Mmx] rnd () == {
       x == uniform_deviate (0.0, 1.0);
       return ball (x, 0.0000001);
     };
```

```
Mmx] M == [ rnd () | j in 1 to 7 || i in 1 to 7 ]
```

$$\begin{bmatrix} 0.092667 & 0.384170 & 0.760549 & 0.966601 & 0.926709 & 0.07336 & 0.709145 \\ 0.536613 & 0.65649 & 0.577904 & 0.037098 & 0.757341 & 0.429381 & 0.53804 \\ 0.64044 & 0.093829 & 0.110069 & 0.33592 & 0.645902 & 0.346289 & 0.825595 \\ 0.175551 & 0.034751 & 0.16514 & 0.651159 & 0.83570 & 0.446781 & 0.942783 \\ 0.085426 & 0.986699 & 0.318042 & 0.860536 & 0.334049 & 0.427479 & 0.846697 \\ 0.116100 & 0.321201 & 0.994810 & 0.012999 & 0.563065 & 0.203569 & 0.813881 \\ 0.363605 & 0.635308 & 0.87561 & 0.31302 & 0.487413 & 0.29065 & 0.853146 \end{bmatrix}$$

```
Mmx] row_echelon M
```

$$\begin{bmatrix} 0.64044 & 0.093829 & 0.110069 & 0.33592 & 0.645902 & 0.346289 & 0.825595 \\ 0e-6 & 0.974184 & 0.303360 & 0.815729 & 0.247895 & 0.381289 & 0.736574 \\ 0e-6 & 0e-6 & 0.880132 & -0.302610 & 0.368569 & 0.021735 & 0.434219 \\ 0e-6 & 0e-6 & 0e-6 & 0.824022 & 0.475454 & -0.13734 & -9.5e-4 \\ 0e-6 & 0e-6 & 0e-6 & 0e-6 & 0.30061 & -0.19834 & -0.74218 \\ 0e-6 & 0e-6 & 0e-6 & 0e-6 & 0e-6 & 0.61384 & 1.27860 \\ 0e-6 & 0e-6 & 0e-6 & 0e-6 & 0e-6 & 0e-6 & -0.23141 \end{bmatrix}$$

3.3 Perte de précision intrinsèque et surestimation

3.3.1 Nombre de conditionnement et perte de précision

En calcul numérique, la précision relative du résultat est généralement moindre que la précision relative des données. On pourrait appeler la différence entre ces deux précisions la « déperdition de précision ». Cette déperdition admet deux sources tout à fait distinctes :

- Une partie de la déperdition est intrinsèque et liée au conditionnement du problème.
- L'autre partie est due à l'algorithme de calcul choisi, et on peut espérer la réduire autant que possible en cherchant un bon algorithme.

Nombre de conditionnement

Fixons une norme $\|\cdot\|$ sur $\mathbb{R}^n$ et soit $f \in \mathbb{R}^n \to \mathbb{R}^k$ une fonction. On définit le *nombre de conditionnement* $\kappa_f(x)$ de f en $x \in \mathbb{R}^n$ par

$$\kappa_f(x) \;=\; \limsup_{\varepsilon \to 0} \frac{\|f(x+\varepsilon) - f(x)\|}{\|f(x)\|} \Big/ \frac{\|\varepsilon\|}{\|x\|}.$$

Le logarithme $\log_2 \kappa_f(x)$ en base deux mesure donc la déperdition intrinsèque de précision pour l'évaluation de f en x. En algèbre linéaire, on définit aussi le nombre de conditionnement d'une matrice par

$$\kappa(M) \;=\; \|M\|\,\|M^{-1}\|$$

Ceci correspond au maximum des $\kappa_f(x)$ pour $x \neq 0$, où $f : x \mapsto M^{-1}\,x$.

Facteur d'éloignement

Supposons maintenant que l'on cherche à approcher f par $f_p\colon \mathbb{D}_p^n \to \mathbb{D}_p^k$, en calculant avec une précision de p bits. On définit le *facteur d'éloignement* de f_p en $x \in \mathbb{D}_p^n$ par

$$\chi_{f_p}(x) \;=\; \frac{1}{\kappa_f(x)}\left(\frac{\|f_p(x) - f(x)\|\, 2^p}{\|f(x)\|} + 1\right)$$

Quand on calcule f_p en x on perd donc environ $\log_2 \kappa_f(x)$ bits de précision de façon intrinsèque et environ $\log_2 \chi_{f_p}(x)$ bits additionnels à cause de l'algorithme employé.

3.3.2 Extensions optimales et surestimation

Essayons maintenant de transposer l'esprit de la section précédente à l'arithmétique de boules. Le nombre de conditionnement n'a pas de contre-partie directe. En revanche, on la notion d'une « extension optimale » :

Extension optimale

Soit $f\colon \mathbb{R}^n \to \mathbb{R}^k$ une fonction. Nous avons déjà signalé l'existence *a priori* d'une multitude d'extensions $f^\bullet\colon \mathcal{B}(\mathbb{R},\mathbb{R})^n \to \mathcal{B}(\mathbb{R},\mathbb{R})^k$ de f vérifiant la relation

$$f(x_1^\bullet, ..., x_n^\bullet) = \{\, f(x_1, ..., x_n) \colon x_1 \in x_1^\bullet, ..., x_n \in x_n^\bullet \,\} \;\subseteq\; f^\bullet(x_1^\bullet, ..., x_n^\bullet)$$

pour tous les $x_1^\bullet, ..., x_n^\bullet \in \mathcal{B}(\mathbb{R},\mathbb{R})^n$. Néanmoins, si on impose la contrainte supplémentaire que $f^\bullet(\mathcal{B}(x_1,r_1), ..., \mathcal{B}(x_n,r_n))$ est de la forme $\mathcal{B}(f(x_1, ..., x_n), s)$ pour un certain $s \in (\mathbb{R}^{\geqslant})^k$, alors il existe une unique *extension optimale* f^*, définie par

$$f^*(\mathcal{B}(x,r)) \;=\; \mathcal{B}\left(\, f(x), \sup_{x' \in \mathcal{B}(x,r)} |f(x') - f(x)| \,\right)$$

Ici, on utilise la notation vectorielle. Par exemple $|(y_1, ..., y_k)| = (|y_1|, ..., |y_k|)$.

Surestimation

Maintenant, nous pouvons définir la surestimation d'une extension $f^\bullet$ arbitraire de façon naturelle par rapport à l'extension optimale f^*. Plus précisément, étant donnée une boule $\mathcal{B}(x,r)$, on définit la *surestimation* de $f^\bullet$ en $\mathcal{B}(x,r)$ par

$$\chi_{f^\bullet}(\mathcal{B}(x,r)) \;=\; \inf_{\lambda \in \mathbb{R}^{\geqslant}} \{\, f^\bullet(\mathcal{B}(x,r)) - f(x) \subseteq \lambda\,(f^*(\mathcal{B}(x,r)) - f(x)) \,\}$$

On définit également la *surestimation ponctuelle* de f en $x \in \mathbb{R}^n$ par

$$\chi_{f^\bullet}(x) \;=\; \operatorname*{limsup}_{r \to 0} \chi_{f^\bullet}(\mathcal{B}(x,r)).$$

Nous allons voir maintenant que cette surestimation ponctuelle est souvent simple à calculer. De la même manière qu'il est bon pour un algorithme numérique de surveiller le nombre de conditionnement et le facteur d'éloignement, c'est généralement une bonne idée pour un algorithme certifié de surveiller la surestimation. En effet, si la surestimation devient trop importante, il est souvent possible de déclencher un autre algorithme *a priori* plus couteux, mais plus précis.

3.3.3 Surestimation de l'arithmétique standard

Supposons que l'on ait une expression f construite à partir de constantes dans $\mathbb{R}$, d'un nombre fini d'indéterminées $X_1, ..., X_n$ et les opérations $+, -, \times$. D'une part, on peut interpréter f comme une fonction $f : \mathbb{R}^n \to \mathbb{R}$. D'autre part, en utilisant l'arithmétique de boules standard (3.2–3.4), l'expression f donne naturellement lieu à une fonction $f^\bullet : \mathcal{B}(\mathbb{R}, \mathbb{R})^n \to \mathcal{B}(\mathbb{R}, \mathbb{R})$. Il n'est pas difficile de montrer que la surestimation ponctuelle de $f^\bullet$ se calcule explicitement par

$$\chi_{f^\bullet}(x) \;=\; \limsup_{\varepsilon \neq 0} \frac{(\bar\nabla\, f)(x) \cdot |\varepsilon|}{|(\nabla f)(x)| \cdot |\varepsilon|} \tag{3.6}$$

où l'opérateur $\bar\nabla$ de « gradient majoré » est défini par

$$\begin{aligned}
\bar\nabla\, c &= (0, ..., 0) && (c \in \mathbb{R}) \\
\bar\nabla\, X_k &= (0, \overset{k-1}{...}, 0, 1, 0, ..., 0) && (k \in \{1, ..., n\}) \\
\bar\nabla(f \pm g) &= \bar\nabla\, f + \bar\nabla\, g \\
\bar\nabla(f g) &= (\bar\nabla\, f)\,|g| + |f|\,(\bar\nabla\, g),
\end{aligned}$$

Lorsque $n = 1$, la formule (3.6) se simplifie en

$$\chi_{f^\bullet}(x) \;=\; \frac{(\bar\nabla\, f)(x)}{|f'(x)|}. \tag{3.7}$$

Exemple 3.3. Prenons l'exemple

$$\begin{aligned}
f(x) &= x^2 - 2\,x + 1 \\
f^\bullet(x^\bullet) &= (x^\bullet)^2 -^\bullet 2\,x^\bullet +^\bullet 1
\end{aligned}$$

L'extension optimale de f est la même que pour la fonction $f(x) = (x-1)^2$:

$$f^*(\mathcal{B}(x, \varepsilon)) \;=\; \mathcal{B}(f(x), 2\,|x-1|\,\varepsilon + \mathcal{O}(\varepsilon^2)),$$

alors que l'extension standard donne

$$f^\bullet(\mathcal{B}(x, \varepsilon)) \;=\; \mathcal{B}(f(x), 2\,|x|\,\varepsilon + 2\,\varepsilon + \mathcal{O}(\varepsilon^2)).$$

Par conséquent,

$$\chi_{f^\bullet}(x) \;=\; \frac{|x| + 1}{|x - 1|},$$

conformément à la formule (3.7).

Cette formule a une conséquence importante pour des algorithmes par subdivision qui cherchent des zéros de f dans un ensemble donné. En effet, lorsque l'on utilise l'arithmétique de boules (ou d'intervalles) naïve pour certifier l'absence de zéros dans des boîtes, on est *forcé* de prendre des boîtes $\chi_{f^\bullet}(x)$ fois plus petites que nécéssaire. En dimension n, cela multiplie le coût de l'algorithme par un facteur $\chi_{f^\bullet}(x)^n$ vis à vis un algorithme théorique optimal. Dans la section 5.4, nous verrons une méthode systématique pour réduire la surestimation afin de remédier à ce problème.

Toujours pour l'arithmétique de boules standard, une problème intéressant est de pouvoir calculer ou au moins estimer la surestimation sur une boule générale par rapport à la surestimation ponctuelle. Voici une question plus précise qui semble abordable :

Question. *Pour $f^\bullet$ construite à partir de $\mathbb{R}, X_1, ..., X_n, +^\bullet, -^\bullet, \times^\bullet$, est-ce que*

$$\chi_{f^\bullet}(\mathcal{B}(x,r)) \leqslant \sup_{x \in \mathcal{B}(x,r)} \chi_{f^\bullet}(x) \ ?$$

3.4 Qualité *versus* efficacité

3.4.1 Le Graal : estimations efficaces et de qualité

Ayant défini la surestimation d'un algorithme de façon précise, la question est maintenant comment construire des algorithmes à la fois efficaces et de bonne qualité (c'est à dire, avec une surestimation proche de 1). Bien évidemment, il s'agit de deux objectifs contraires, donc il faudra chercher un compromis.

Par exemple, en rendant systématiquement la boule $\mathcal{B}(0, \infty)$ comme résultat, on obtient un algorithme très efficace, mais sans intérêt. Réciproquement, dans de nombreux cas, l'obtention de bornes optimales est NP complet ou pire [44]. Généralement, les algorithmes qui nous intéressent introduisent un facteur constant dans la complexité pour calculer des bornes de qualité acceptable, ou des petits facteurs logarithmiques ou polynomiaux pour avoir un algorithme avec une surestimation proche de 1.

On rappelle aussi qu'il est souvent possible d'appliquer la stratégie de la « terminaison précoce » : on commence avec un algorithme rapide et *a priori* de faible qualité. Seulement si la qualité du résultat est jugée insuffisante, on lance des algorithmes plus lents, mais de meilleure qualité.

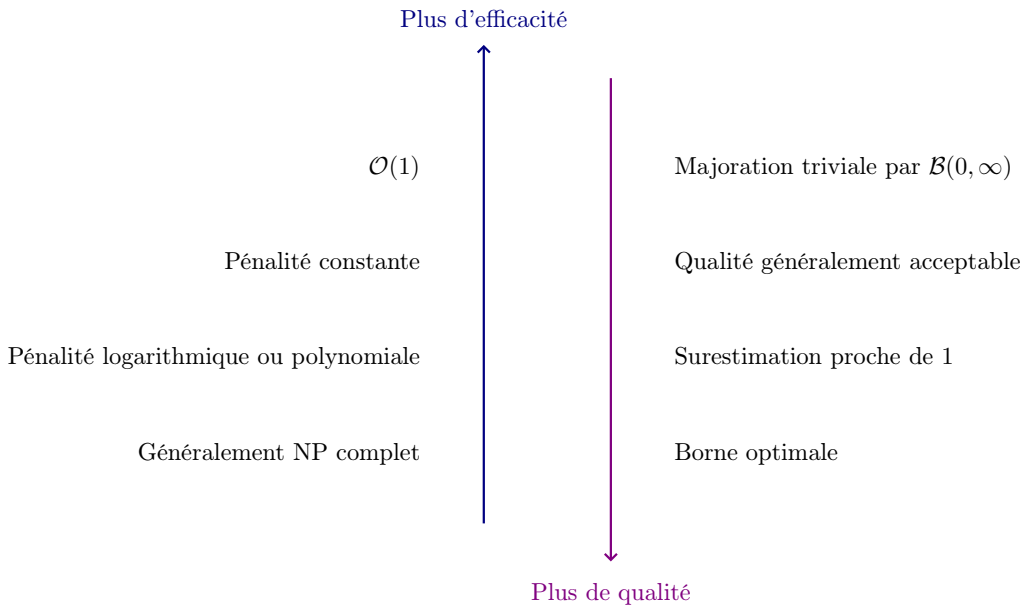


Fig. 3.2. Recherche d'un compromis entre efficacité et qualité des bornes d'erreur.

3.4.2 Multiplication de matrices

La recherche d'un compromis entre efficacité et qualité s'illustre bien sur le problème de la multiplication de deux matrices $M^\bullet, N^\bullet \in \mathcal{B}(\mathbb{D}, \mathbb{D})^{n\times n}$.

Stratégie naïve

On calcule le produit par l'algorithme classique en utilisant $\mathcal{O}(n^3)$ opérations en arithmétique de boules. C'est un algorithme de bonne qualité, mais assez lent.

Emploi de boules matricielles [62]

On réinterprète $M^\bullet$, $N^\bullet \in \mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$ comme des boules avec des centres et des rayons dans $\mathbb{D}^{n\times n}$. Dès lors, on peut utiliser la formule (3.4) pour la multiplication, quitte à rajouter le nécessaire pour les erreurs d'arrondi.

L'avantage principal de cette méthode est que l'on s'appuie directement sur la multiplication matricielle dans $\mathbb{D}^{n\times n}$ au lieu de faire toutes les opérations sur les coefficients un par un dans l'arithmétique des boules. En effet, la multiplication dans $\mathbb{D}^{n\times n}$ est généralement hautement optimisée (en précision machine, on peut par exemple utiliser des BLAS), donc on gagne au moins une grosse constante par rapport à la stratégie naïve.

Majorations brutales

Dans la méthode précédente, on majore le rayon du produit par la somme de deux produits de matrices dans $(\mathbb{D}^{\geqslant})^{n\times n}$. Étant données deux matrices $A, B \in (\mathbb{D}^{\geqslant})^{n\times n}$, il est souvent possible d'obtenir une majoration raisonnable du produit $A\,B$ en faisant seulement $\mathcal{O}(n^2)$ opérations.

En effet, supposons qu'après préconditionnement, on peut s'arranger pour que les entrées de chaque ligne de A et de chaque colonne de B soient du même ordre de grandeur. Dans ce cas, on peut utiliser la formule

$$|(A\,B)_{ij}| \;\leqslant\; [\max |A_{i\cdot}|]\,[\max |B_{\cdot j}|].$$

Autres compromis

Dans la pratique, il arrive souvent que les entrées d'une matrice soient localement du même ordre de grandeur, mais pas globalement, même après préconditionnement. Dans ce cas, on peut découper la matrice en petits blocs de matrices $k \times k$, utiliser des majorations brutales sur ces petits blocs et un algorithme plus naïf pour la matrice toute entière. Ceci conduit à un algorithme de coût C_k supérieur à la multiplication dans $\mathbb{D}^{n\times n}$, avec $1 \leqslant C_k \leqslant 3$.

De manière générale, on observe aussi que le coût de certification des calculs tend souvent à devenir négligeable pour des gros problèmes. Par exemple, pour la multiplication dans $\mathcal{B}(\mathbb{D}_p, \mathbb{D}_p)$, l'estimation des erreurs se fait généralement en simple précision. En grande précision, le coût de deux multiplications en simple précision devient négligeable devant le coût d'une grosse multiplication dans $\mathbb{D}_p$. De même, dans des cas bien conditionnés où on peut utiliser la méthode des majorations brutales pour la multiplication des matrices, le coût des $\mathcal{O}(n^2)$ opérations pour estimer les erreurs est négligeable devant le coût $\mathcal{O}(n^3)$ pour la multiplication des centres.

3.5 Hiérarchie numérique

À présent, nous avons vu les types de base pour le calcul analytique. D'un point de vue haut niveau, nous avons les nombres calculables de $\mathbb{R}^{\mathrm{cal}}$, $\mathbb{R}^{\mathrm{calg}}$, $\mathbb{R}^{\mathrm{cald}}$, etc. Juste en-dessous suit l'arithmétique de boules qui permet le calcul certifié avec des approximations. Tout en bas, nous avons le calcul numérique classique avec des nombres en virgule flottante et l'arithmétique rapide pour les mantisses en précision arbitraire.

Cette division en quatre niveaux de la « hiérarchie numérique » est pertinente pour la plupart des algorithmes en calcul analytique. Dans le cas d'une multiplication de matrices par exemple, on procède comme suit :

- Au niveau conceptuel, nous partons de deux matrices $M, N \in (\mathbb{R}^{\mathrm{cal}})^{n\times n}$ à multiplier. Une telle matrice est constituée de n^2 promesses d'approximation. Pour gagner en efficacité il vaut mieux réécrire $M, N \in (\mathbb{R}^{n\times n})^{\mathrm{cal}}$.
- Pour une précision p donnée, on approche M et N par des matrices de boules $M^\bullet$, $N^\bullet \in \mathcal{B}(\mathbb{D}, \mathbb{D})^{n\times n}$, ou mieux, encore pour des raisons d'efficacité, comme des boules matricielles $M^\bullet, N^\bullet \in \mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$.
- L'arithmétique de boules dans $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$ conduit à l'arithmétique en virgule flottante standard dans $\mathbb{D}^{n\times n}$ sur les centres et les rayons. Si p est petit, on peut utiliser des nombres machines et des BLAS rapides.
- Si p est plus grand, et si on cherche à multiplier efficacement des matrices $\tilde{M}, \tilde{N} \in \mathbb{D}^{n\times n}$, il est souvent possible de les préconditionner et ensuite à les écrire par rapport à un même exposant : $\tilde{M}, \tilde{N} \in \mathbb{Z}^{n\times n}\, 2^{\mathbb{Z}}$. On peut alors employer un algorithme de multiplication de matrices rapide dans $\mathbb{Z}^{n\times n}$.

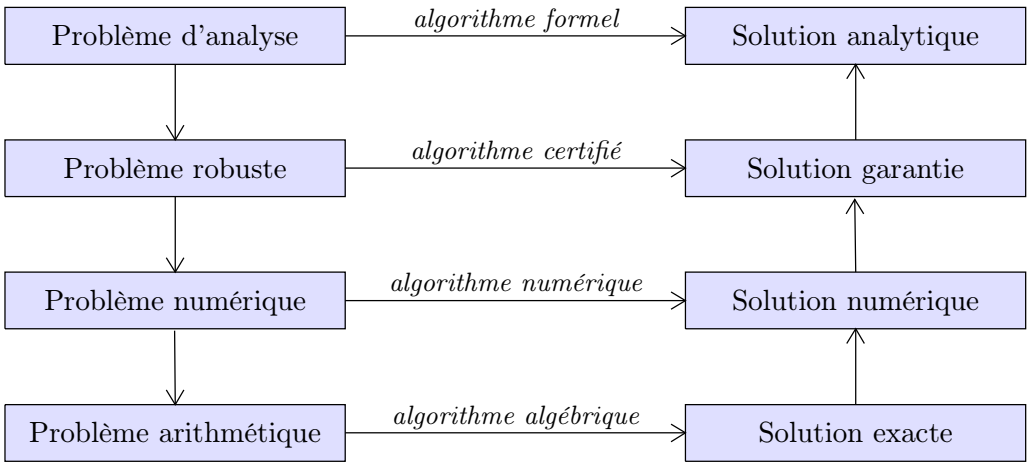


Fig. 3.3. La hiérarchie numérique.

4

Arithmétique rapide

4.1 Rappels sur la complexité

L'algorithmique rapide est un sujet classique en calcul formel. Nous ferons quelques rappels, sans chercher à être exhaustif. Voir par exemple [31, 6] pour deux références classiques.

L'opération clef à comprendre pour des analyses en complexité est la multiplication, que ça soit sur les entiers, sur les nombres flottants, les polynômes, les matrices, ou encore les séries. Les complexités d'autres opérations (division, racine carrée, etc.) s'expriment généralement en fonction du coût de la multiplication. Voici quelques complexités classiques concernant la multiplication :

Multiplication de deux entiers de p chiffres

- $\mathsf{I}(p) = \mathcal{O}(p^2)$: multiplication naïve.
- $\mathsf{I}(p) = \mathcal{O}\big(p^{\log 3/\log 2}\big)$: multiplication de Karatsuba [41].
- $\mathsf{I}(p) = \mathcal{O}(p \log p \log\log p)$: multiplication de Schönhage-Strassen [67].
- $\mathsf{I}(p) = \mathcal{O}(p \log p\, 2^{\log^* p})$: multiplication de Fürer [30].

Multiplication de deux polynômes à coefficients dans $\mathbb{K}$ de degré n

- $\mathsf{M}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}(n^2)$: multiplication naïve (on compte le nombre d'opérations dans $\mathbb{K}$).
- $\mathsf{M}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}\big(n^{\log 3/\log 2}\big)$: multiplication de Karatsuba [41].
- $\mathsf{M}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}(n \log n \log\log n)$: multiplication de Schönhage-Strassen [67, 17].
- $\mathsf{M}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}(n \log n)$ si $\mathbb{K}$ admet suffisamment de racines 2^p-ièmes de l'unité [19].

Multiplication de deux matrices dans $\mathbb{K}^{n\times n}$

- $\mathcal{O}_{\mathbb{K}}(n^3)$: multiplication naïve.
- $\mathcal{O}_{\mathbb{K}}(n^{\log_2 7})$: multiplication de Strassen [74].
- $\mathcal{O}_{\mathbb{K}}(n^{\omega})$ avec $\omega < 2.376$: multiplication de Coppersmith et Winograd [58, 20].

Multiplication détendue de deux séries dans $\mathbb{K}[[z]]$ à l'ordre n (voir la section 4.3)

- $\mathsf{R}_{\mathbb{K}}(n) \sim \frac{1}{2}\,\mathsf{M}_{\mathbb{K}}(n)$: multiplication naïve.

- $\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}\big(n^{\log 3/\log 2}\big)$: multiplication de Karatsuba détendue [76, 79].
- $\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}(\mathsf{M}_{\mathbb{K}}(n)\log n)$: multiplication détendue rapide [76, 79].
- $\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}_{\mathbb{K}}\Big(\mathsf{M}_{\mathbb{K}}(n)\ \mathrm{e}^{2\sqrt{\log\log n}}\Big)$ si $\mathbb{K}$ admet suffisamment de racines 2^p-ièmes de l'unité [83].

Multiplication de deux polynômes de degré n avec des coefficients de p chiffres

- $\mathsf{IM}(p,n) = \mathcal{O}(\mathsf{I}(n\,(p+\log n)))$: Kronecker [31].

Multiplication de deux polynômes de degré d avec des coefficients dans $\mathbb{K}^{n\times n}$

- $\mathsf{MM}(d,\,n) = \mathcal{O}_{\mathbb{K}}(d\ n^{\omega} + \mathsf{M}_{\mathbb{K}}(d)\ n^2)$ si la complexité $\mathsf{M}_{\mathbb{K}}(d)$ pour la multiplication est obtenue par évaluation-interpolation en $\mathcal{O}(d)$ de points.

4.2 Algorithmes sur les polynômes et sur les séries

4.2.1 Multiplication de polynômes

Multiplication de Karatsuba

Il existe deux variantes pour multiplier deux polynômes $P,\,Q \in \mathbb{K}[z]$ de degrés $<n$. On peut couper les polynômes en deux à l'exposant $\lfloor n/2 \rfloor$, ou par exposants pairs-impairs. L'algorithme pair-impair est généralement plus stable d'un point de vue numérique :

$$\begin{aligned} & (P_{\mathbb{O}} + P_{\mathbb{I}}\,z)\,(Q_{\mathbb{O}} + Q_{\mathbb{I}}\,z) \\ = \ & P_{\mathbb{O}}\,Q_{\mathbb{O}} + ((P_{\mathbb{O}} + P_{\mathbb{I}})\,(Q_{\mathbb{O}} + Q_{\mathbb{I}}) - P_{\mathbb{O}}\,Q_{\mathbb{O}} - P_{\mathbb{I}}\,Q_{\mathbb{I}})\,z + P_{\mathbb{I}}\,Q_{\mathbb{I}}\,z^2 \end{aligned}$$

Multiplication FFT

La multiplication de Karatsuba est un algorithme « multi-modulaire » classique, procédant par évaluation-interpolation dans les points $z = 0, 1, \infty$ (en projectif). Si $\mathbb{K}$ admet une racine primitive de l'unité ω d'ordre $n = 2^p \geqslant 2$ (donc $\omega^{n/2} = 1$), une stratégie multi-modulaire plus efficace est basée sur la transformation de Fourier discrète rapide (FFT). Pour un polynôme $P \in \mathbb{K}[z]$ avec $\deg P < n$, on définit $\mathrm{FFT}_\omega(P) \in \mathbb{K}^n$ par

$$\mathrm{FFT}_\omega(P) \ = \ (P(1), P(\omega), ..., P(\omega^{n-1}))$$

et il est classique [19] que l'on peut calculer FFT_ω et son inverse FFT_ω^{-1} en temps $\mathcal{O}_{\mathbb{K}}(n\,\log\,n)$. Étant donné deux polynômes $P,\,Q \in \mathbb{K}[z]$ avec $\deg\,P\,Q < n$ on peut donc calculer leur produit en temps $\mathcal{O}(n\log n)$ par la formule

$$PQ \ = \ \mathrm{FFT}_\omega^{-1}(\mathrm{FFT}_\omega(P)\,\mathrm{FFT}_\omega(Q)).$$

4.2.2 Multiplication de séries tronquées

Un point de vue naturel pour calculer avec des séries dans $\mathbb{K}[[z]]$ est de calculer systématiquement avec des séries à l'ordre $\mathcal{O}(z^n)$. Pour multiplier deux telles séries, il suffit de multiplier les séries tronquées en tant que polynômes et de tronquer le résultat. Toutefois, si on utilise la multiplication naïve, ceci conduit à faire environ deux fois trop de travail. C'est un problème ouvert de savoir si on peut faire mieux en général :

Question. *Est-ce qu'il existe un algorithme de multiplication tronquée avec de complexité en temps $\lambda\,\mathsf{M}_{\mathbb{K}}(n)$ avec $\lambda < 1$?*

4.2.3 Méthode de Newton

On a déjà signalé le fait que la complexité de la plupart des opérations plus complexes, comme la division, la racine carrée, l'exponentielle, etc. s'expriment en fonction de la complexité de la multiplication. Une technique puissante pour démontrer ceci est la méthode de Newton.

Supposons par exemple que l'on veuille inverser une série $f \in \mathbb{K}[[z]]$ avec $f_0 \neq 0$ à l'ordre n. En d'autres mots, étant donnés les coefficients $f_0, \ldots, f_{n-1}$, on voudrait calculer les coefficients $g_0, \ldots, g_{n-1}$ de $g = f^{-1}$. La méthode de Newton appliquée à l'équation $f\,g - 1 = 0$ donne l'itération

$$\tilde{g} \;=\; g - \frac{f\,g - 1}{f}.$$

Comme on ne connait pas encore $1/f$ dans cette itération, on utilisera plutôt l'itération

$$\tilde{g} \;=\; g - (f\,g - 1)\,g.$$

Si g est bon jusqu'à l'ordre n :

$$g \;=\; \frac{1}{f} + \mathcal{O}(z^n),$$

la nouvelle valeur $\tilde{g}$ sera bonne jusqu'à l'ordre $2\,n$, puisque

$$\begin{aligned} \tilde{g} \;&=\; g - (f\,g - 1)\,g \\ &=\; g - \frac{f\,g - 1}{f} + \mathcal{O}(z^{2n}) \\ &=\; \frac{1}{f} + \mathcal{O}(z^{2n}) \end{aligned}$$

Si $T(n)$ désigne le coût de la division à l'ordre n, ceci donne

$$T(n) \;=\; T(n/2) + \mathcal{O}(\mathsf{M}_{\mathbb{K}}(n)).$$

En supposant que $\mathsf{M}_{\mathbb{K}}(n)/n$ est croissante, on laisse au soin du lecteur de vérifier que ceci implique $T(n) = \mathcal{O}_{\mathbb{K}}(\mathsf{M}_{\mathbb{K}}(n))$. On déduit aussi aisément que le calcul du quotient et du reste de la division euclidienne d'un polynôme P de degré $2\,n$ par un polynôme Q de degré n se calcule en temps $\mathcal{O}_{\mathbb{K}}(\mathsf{M}_{\mathbb{K}}(n))$.

L'utilisation de la méthode de Newton dans ce cadre remonte à Hensel et se généralise pour des équations polynomiales plus générale (à la fois dans les séries et les nombres p-adiques d'ailleurs). Brent et Kung ont remarqué [13, 15] que la même technique peut être utilisée pour composer des séries formelles, calculer leurs inverses fonctionnelles, ou résoudre des équations différentielles. Par exemple, l'exponentielle $g = \exp f$ de $f \in \mathbb{K}[[z]]$ avec $f_0 = 0$ se calcule par l'itération

$$\begin{aligned} \tilde{g} &= g - g\,(\log g - f) \\ \log g &= \int \tfrac{g'}{g}. \end{aligned}$$

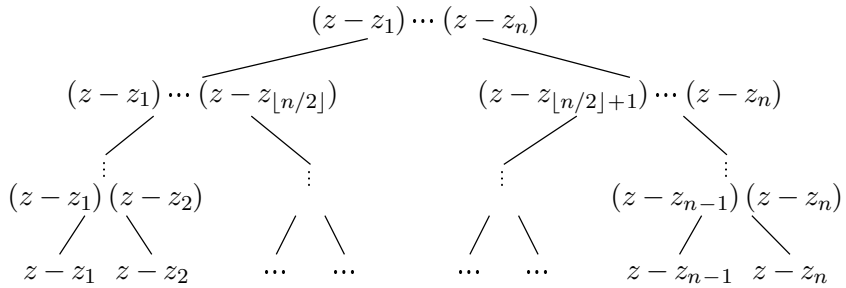
Une méthode efficace pour la résolution d'équations différentielles algébriques plus générales a été proposée par Sedoglavic [68, 10, 86].

4.2.4 Évaluation multi-points

Il y a de nombreux autres opérations sur les polynômes (racine carrée, pgcd, ppcm, etc.) qui peuvent se calculer en temps presque linéaire. Nous donnons un dernier exemple qui est utile pour le calcul des zéros d'un polynôme. En effet, si on dispose déjà de bonnes approximations pour les racines, l'évaluation multi-point rapide permet d'appliquer la méthode de Newton pour améliorer toutes ces approximations de façon simultanée.

Algorithme évaluation multi-point rapide
Entrées : $P \in \mathbb{K}[z]$ avec $\deg P < n$ et points $z_1, \ldots, z_n \in \mathbb{K}$
Sortie : $P(z_1), \ldots, P(z_n) \in \mathbb{K}$

- On précalcule tous les polynômes dans l'arbre suivant :



- On utilise la méthode dichotomique suivante pour calculer le résultat :
  - Si $n = 1$, retourner P_0
  - Sinon, soient $Q = (z-z_1)\cdots(z-z_{\lfloor n/2\rfloor})$ et $R = (z-z_{\lfloor n/2\rfloor+1})\cdots(z-z_n)$
  - Evaluer $P \operatorname{rem} Q$ en $z_1, \ldots, z_{\lfloor n/2\rfloor}$ et $P \operatorname{rem} R$ en $z_{\lfloor n/2\rfloor+1}, \ldots, z_n$

On remarque que la complexité du précalcul et de la dichotomie proprement dite vérifient une estimation du type

$$T(n) = 2\,T(n/2) + \mathcal{O}(\mathsf{M}_{\mathbb{K}}(n)).$$

En supposant que $\mathsf{M}_{\mathbb{K}}(n)/n$ est croissante, on laisse au soin du lecteur de vérifier que ceci implique $T(n) = \mathcal{O}(\mathsf{M}_{\mathbb{K}}(n) \log n)$. On peut encore gagner un facteur constant sur l'algorithme présenté ici [11].

4.3 Calcul détendu

4.3.1 Principe du calcul détendue

Nous avons vu comment calculer avec des séries en les tronquant à un certain ordre n. Une autre approche consiste à considérer les séries comme des « flots de coefficients » qui sont calculés un par un. Ce point de vue détendu impose une contrainte sur notre façon de concevoir les opérations sur les séries. Par exemple, la multiplication détendue $h = f\,g$ de deux séries $f, g \in \mathbb{K}[[z]]$ impose la sortie de h_n *dès que* $f_0, g_0, \ldots, f_n, g_n$ sont connus. Par conséquent, on ne peut pas directement appliquer certains algorithmes rapides, comme la multiplication FFT. L'avantage du calcul détendu est qu'il permet *naturellement* de résoudre des équations « récursives »

$$f \;=\; \Phi(f),$$

où l'extraction du coefficient en z^n de $\Phi(f)$ ne fait intervenir que les coefficients $f_0, \ldots, f_{n-1}$ de f. Voici deux exemples :

Inverse g de $1 - f$ avec $f \in z\,\mathbb{K}[[z]]$

$$\begin{aligned} g &= 1 + f\,g \\ g_n &= \sum_{k=0}^{n-1} f_{n-k}\,g_k, \qquad n > 0. \end{aligned}$$

Exponentielle g de f avec $f \in z\,\mathbb{K}[[z]]$

$$\begin{aligned} g &= 1 + \int f'\,g \\ g_n &= \frac{1}{n} \sum_{i=0}^{n-1} (n-i)\,f_{n-i}\,g_i, \qquad n > 0. \end{aligned}$$

4.3.2 Multiplication détendue naïve

La stratégie la plus naïve pour calculer le coefficient h_n d'un produit $h = f\,g$ est d'utiliser la formule classique de convolution

$$h_n \;=\; f_0\,g_n + \cdots + f_n\,g_0.$$

Pour un calcul jusqu'à l'ordre n, cela donne une complexité $\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}(n^2)$. Dans la figure 4.1, on montre les étapes successives du calcul détendu de l'exponentielle $\exp \frac{z}{1-z}$.

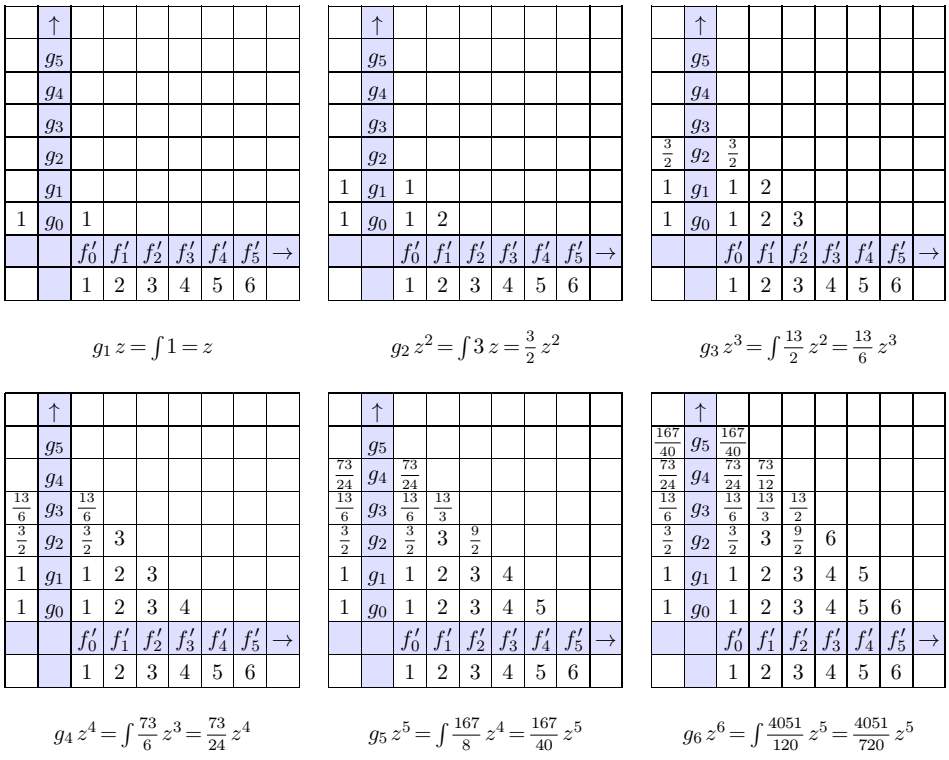


Fig. 4.1. Exponentiation $g = \exp \frac{z}{1-z} = \int f'\, g$ par multiplication détendue de f' et g.

4.3.3 Multiplication détendue par Karatsuba

Lorsque l'on applique l'algorithme de Karatsuba sur des polynômes $f = f_0 + \cdots + f_{n-1}\, z^{n-1}$ et $g = g_0 + \cdots + g_{n-1}\, z^{n-1}$ avec $n = 2^p \geqslant 2$, où on considère $f_0, g_0, ..., f_{n-1}, g_{n-1}$ comme des paramètres formels, on observe que la formule pour $(f\, g)_k$ ne dépend que de $f_0, g_0, ..., f_k, g_k$ pour chaque k. En d'autres termes, l'algorithme de Karatsuba est naturellement détendu, si on fait les calculs dans le bon ordre. Ceci montre qu'il existe un algorithme de multiplication détendue de complexité $\mathsf{R}_{\mathbb{K}}(n) = \mathsf{T}_{\mathbb{K}}(n)$.

4.3.4 Multiplication détendue rapide

On peut encore faire mieux en anticipant des calculs à venir. Par exemple lorsque f_0, f_1, f_2 et g_0, g_1, g_2 sont connus, on peut calculer la contribution de $(f_1\, z + f_2\, z^2)\, (g_1\, z + g_2\, z^2) = (f_1 + f_2\, z)\, (g_1 + g_2\, z)\, z^2$ au produit $f\, g$ par n'importe quel algorithme de multiplication rapide pour les polynômes. De même, lorsque l'on connait f_0, ..., f_6 et g_0, ..., g_6, on peut calculer la contribution de $(f_3 + f_4\, z + f_5\, z^2 + f_6\, z^3)\, (g_3 + g_4\, z + g_5\, z^2 + g_6\, z^3)\, z^6$ au produit $f\, g$ de façon rapide. En exploitant cette idée, les contributions de tous les grands carrés dans la figure 4.2 peuvent se calculer par un algorithme rapide. Il s'ensuit [76, 79] que

$$\mathsf{R}_{\mathbb{K}}(n) \;=\; \mathcal{O}_{\mathbb{K}}(\mathsf{M}_{\mathbb{K}}(n) \log n)$$

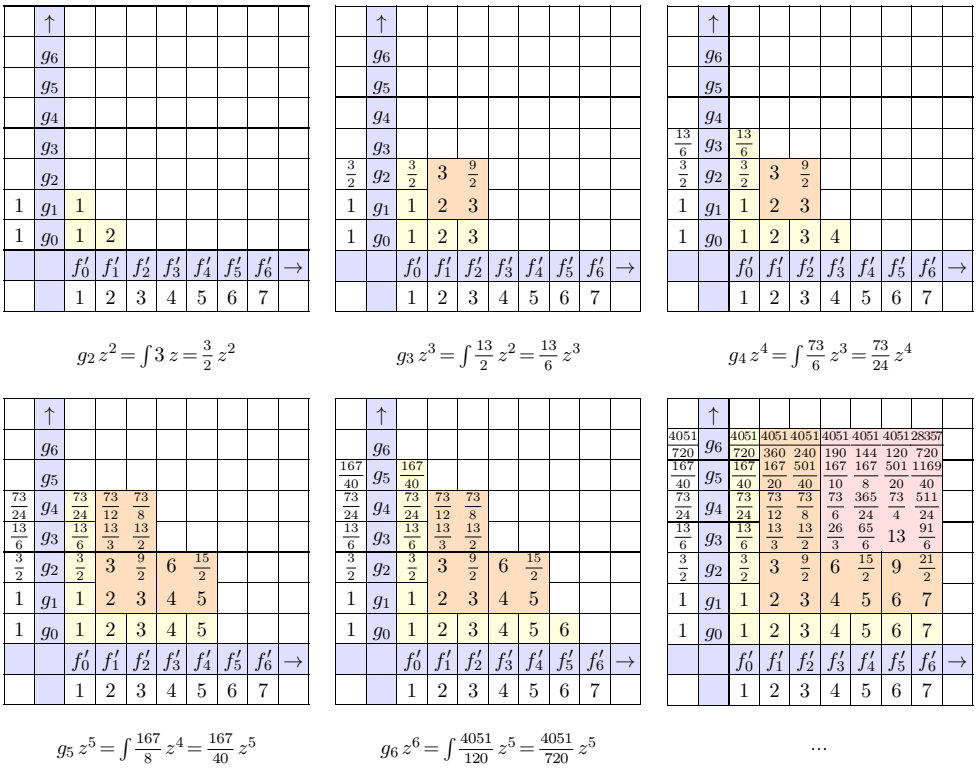


Fig. 4.2. Exponentiation $g = \exp\frac{z}{1-z} = \int f'\,g$ par multiplication détendue rapide de f' et g.

4.3.5 Multiplication détendue super rapide

Est-ce que l'on peut faire encore mieux ? Si $\mathbb{K}$ admet suffisamment de racines 2^p-ièmes de l'unité, on montre [83] que :

$$\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}\Big(\mathsf{M}_{\mathbb{K}}(n)\,\mathrm{e}^{2\sqrt{\log\log n}}\Big).$$

Dans le cas général, la question reste ouverte :

Question. *Pour n'importe quel anneau effectif $\mathbb{K}$, est-ce qu'il existe un algorithme de multiplication détendue dont la complexité est meilleure que*

$$\mathsf{R}_{\mathbb{K}}(n) = \mathcal{O}(\mathsf{M}_{\mathbb{K}}(n)\log n)\ ?$$

4.3.6 Exemple : nombres de Bell exacts

```
Mmx] use "algebramix"
Mmx] z == series (0, 1);
Mmx] B == exp (exp z - 1)
```

$$1 + z + z^2 + \frac{5}{6}\,z^3 + \frac{5}{8}\,z^4 + \frac{13}{30}\,z^5 + \frac{203}{720}\,z^6 + \frac{877}{5040}\,z^7 + \frac{23}{224}\,z^8 + \frac{1007}{17280}\,z^9 + O(z^{10})$$

```
Mmx] B[1000] * 1000!
```

29899013356824084214804223538976464839473928098212305047832737888945413625\
12325959664116587254039157830063914708298696402802180224899338288101341127\
65748291211558117551708306660398388372739719716767823898008103618093192507\
55399325279656765435255999301529770267107281619733800281695881540007577899\
10687867945116549253593045923371331634255154524281580236725728485261220108\
10163863085359901454473418004554723347138640805239789602963657369992959320\
80550928561633025800627524911700149562106895897725047744775812241800937310\
49179781007578233924187312824632629095993832334781713007323483688294825326\
89745368173274105329250746138883212641380838421962022429560013149534494972\
44271843921739082521076522013469338897410704353506902420620015226978552783\
56012043992855156781339712541914478047647919799092160201587370382076918260\
38367554718392851567813397125419144780476479197990921602015873703820769182\
60383678846578509356368602569026980215380243687353087700673715452389527302\
95102387459973562922326312827737487629893860039702144238439470940211779897\
37557020369751561595003372955621411858485959813344799967960196238368337022\
34694677176980306026928869169402844479120397853345475941058706502254649151\
88712384215608259071358856192217764058987710572705555814492299942157394767\
58787858845457230622639923677500913196448615476584722822840058920443715875\
60711880627741139497818835632120761570174928529697397267899554407350161283\
09712321104804926972765527978390070241609513282776642886501765336669630413\
14366902329794538763375997217728970492702305442626112649173933747563841527\
84943607952408782612639220380791445272655004475989064276373713608901650681\
16546749031089880491682706942731096110928503554508479133942326648235995566\
33772015152043408175809154684899691816433410071978364814610517989956407892\
92580146918580703759556640194519451731530034209189203377522668309771129566\
10810161772744204563709811267886543099877854633073765443395068782672673493\
48171320834971956806668304099159992067385998690820326902473886782781499414\
773179
```

## 4.4 Méthodes multi-modulaires

### 4.4.1 Principes et variantes

La multiplication FFT de la section 4.2.1 est un exemple classique d'algorithme multi-modulaire. En effet, elle repose en réalité sur l'isomorphisme

$$\mathbb{K}[z]/(z^n-1) \;\cong\; \prod_{k=0}^{n-1} \mathbb{K}[z]/(z-\omega^k) \tag{4.1}$$

entre les polynômes $P$ modulo $z^n - 1$ et les réductions de $P$ modulo les $z - \omega^k$ avec $k \in \{0, ..., n - 1\}$. De la même manière, étant donnés $l$ nombres premiers $p_1$, ..., $p_l$ mutuellement distincts, le théorème des restes chinois nous fournit un isomorphisme

$$\mathbb{Z}/p_1 \cdots p_l\,\mathbb{Z} \;\cong\; \mathbb{F}_{p_1} \times \cdots \times \mathbb{F}_{p_l}. \tag{4.2}$$

Dans chaque cas, le calcul multi-modulaire repose sur un changement de représentation, où un objet dans $\mathbb{K}[z]/(z^n-1)$ ou $\mathbb{Z}/p_1 \cdots p_l\,\mathbb{Z}$ où la multiplication coûte cher est remplacé par un vecteur d'objets pour lesquels la multiplication coûte moins cher.
```

Comme les changements de représentation coûtent généralement cher aussi, il est recommandé d'effectuer un maximum de calculs dans le modèle multi-modulaire. Supposons par exemple que l'on veuille multiplier deux matrices M et N dans $[\mathbb{K}[z]/(z^d-1)]^{n\times n}$. Au lieu de faire n^ω opérations dans $\mathbb{K}[z]/(z^d-1)$, de coût $\mathcal{O}(\mathsf{M}_{\mathbb{K}}(d)\,n^\omega)$, il vaut mieux utiliser la formule

$$M\,N \;=\; \mathrm{FFT}_z^{-1}(\mathrm{FFT}_z(M)\,\mathrm{FFT}_z(N)),$$

qui correspond à mettre tous les coefficients de M et N dans le modèle FFT, de multiplier d matrices scalaires dans $\mathbb{K}^{n\times n}$, et de faire la transformation inverse. En effet, cet algorithme admet une complexité $\mathcal{O}_{\mathbb{K}}(d\,n^\omega + d\log d\,n^2)$ bien meilleure.

Il y a plusieurs variantes pour les méthodes multi-modulaires. Chaque variante admet ses caractéristiques propres quant au nombre de moduli utilisés et quant au coût de la réduction et de la reconstruction. Les méthodes les plus utilisées sont les suivantes :

Restes chinois

Cette méthode permet *grosso modo* d'encoder un entier d'environ l mots machines en utilisant seulement l moduli. En revanche, les coûts de la réduction et de la reconstruction sont en $\mathcal{O}(\mathsf{I}(p)\log p)$ pour un entier de taille p.

FFT sur $\mathbb{F}_p$ avec beaucoup de racines 2^k-ièmes de l'unité

Certains nombres premiers comme $p = 3\cdot 2^{30}+1$ admettent des racines primitives de l'unité d'un ordre 2^k avec k grand. Cela permet d'utiliser la multiplication FFT pour des polynômes de degrés importants à coefficients dans $\mathbb{F}_p$. Par ailleurs, tout nombre dans $\mathbb{Z}$ peut se coder par $n = P_0 + \cdots + P_d\,p^d$ avec $|P_i| < p/2$. En prenant $|P_i| \ll \sqrt{p}$, on peut aussi ramener la multiplication dans $\mathbb{Z}$ à la multiplication dans $\mathbb{F}_p[z]$ pour des nombres pas trop grands. Par rapport aux restes chinois, ceci revient à prendre plus de moduli, mais aussi à gagner sur le coûts de réduction et de reconstruction. On peut d'ailleurs combiner les deux approches.

FFT numérique

Lorsque l'on calcule avec des nombres flottants complexes, on peut directement calculer avec une racine 2^p-ième de l'unité approchée. En encadrant les erreurs d'arrondi avec soin, ceci conduit à des algorithmes de multiplication rapide dans $\mathbb{D}[z]$ et $\mathbb{Z}$ qui ont des avantages et inconvénients semblables par rapport à la méthode précédente.

FFT synthétique de Schönhage-Strassen

En cas d'absence de racines 2^k-ièmes de l'unité pour k suffisamment grand, on peut les *synthétiser*. Supposons par exemple, que l'on veuille multiplier des polynômes dans $\mathbb{K}[z]/(z^n+1)$ et prenons $m = 2^k \geqslant 2\sqrt{n}$. Alors z est une racine 2^{k+1}-ième primitive de l'unité dans $\mathbb{K}[z]/(z^m+1)$ et montre que la multiplication dans $\mathbb{K}[z]/(z^n+1)$ se ramène à une multiplication de polynômes à coefficients dans $\mathbb{K}[z]/(z^m+1)$. En outre, la FFT à coefficients dans $\mathbb{K}[z]/(z^m+1)$ est très efficace, car elle ne nécessite que des additions et des soustractions dans $\mathbb{K}$. Par ailleurs, la méthode est parfaitement générale, et marche encore pour des grands entiers. Le seul point noir de l'approche est qu'elle nécessite plus de moduli que les méthodes précédentes.

4.4.2 Évaluation rapide de dags

Un problème qui revient fréquemment est l'évaluation rapide d'un vecteur de polynômes en plusieurs variables $P \in \mathbb{K}[X_1, ..., X_d]^r$ dans une $\mathbb{K}$-algèbre $\mathbb{A}$. En effet, ce problème intervient dans l'intégration de système dynamiques $Y' = P(Y)$ ou encore dans des méthodes algébriques par homotopie. En outre, il n'est pas rare que l'on veuille évaluer à la fois P et sa matrice jacobienne $\partial P/\partial X$.

Nous allons nous concentrer sur le cas $\mathbb{A} = \mathbb{K}[z]/(z^n)$, en supposant que la multiplication dans $\mathbb{A}$ s'effectue par un algorithme multi-modulaire avec $\mathsf{N}(n)$ moduli et avec un coût de réduction-reconstruction de $\mathsf{E}_{\mathbb{K}}(n)$ opérations dans $\mathbb{K}$.

Considérons d'abord le cas où les polynômes sont donnés par des expressions avec éventuellement des sous-expressions communes. En d'autres termes, P est représenté par un « dag » (*directed acyclic graph*). La figure 4.3 montre un dag typique, correspondant au vecteur (P_1, P_2) avec

$$\begin{aligned} P_1 &= (X_1 - 2)^2 (X_1 + X_2)^2, \\ P_2 &= (X_1 + X_2)^2 + (X_1 + X_2)(3\, X_2). \end{aligned}$$

Un sous-dag de P est dit *scalaire* s'il ne fait pas intervenir les $X_1, ..., X_d$. Dans la figure 4.3, les seuls sous-dags scalaires sont 2 et 3. Un sous-dag *multiplicatif* de P de la forme $A \times B$ est dit

homothétique, si A ou B est scalaire ;
terminal, si $A \times B$ n'est pas un sous-dag d'un autre sous-dag multiplicatif de P ;
actif, dans les cas restants.

Dans notre exemple, les ensembles de sous-dags multiplicatifs homothétiques, terminaux et actifs sont respectivement $\{3\, X_2\}$, $\{(X_1 + X_2)(3\, X_2), P_1\}$ et $\{(X_1 - 2)^2, (X_1 + X_2)^2\}$. On a :

Proposition 4.1. *Soit $P \in \mathbb{K}[X_1, ..., X_d]^r$ un dag de taille totale t et avec μ sous-dags multiplicatifs actifs. Alors pour des séries tronquées $f_1, ..., f_d \in \mathbb{K}[z]/(z^n)$, on peut évaluer $P(f_1, ..., f_d)$ en utilisant au plus $(r + d + 2\,\mu)\, \mathsf{E}_{\mathbb{K}}(n) + (t - d)\, \mathsf{N}(n)$ opérations dans $\mathbb{K}$.*

Démonstration. On commence par réduire les entrées $f_1, ..., f_d$ (coût $d\, \mathsf{E}_{\mathbb{K}}(n)$). De façon récursive, et travaillant avec la représentation multi-modulaire, on évalue chaque sous-dag non multiplicatif ou non actif en utilisant $\mathsf{N}(n)$ opérations dans $\mathbb{K}$. On évalue chaque sous-dag multiplicatif actif en faisant un produit scalaire en représentation multi-modulaire (coût $\mathsf{N}(n)$), une reconstruction d'un polynôme dans $\mathbb{K}[z]$ de degré $<(2\, n - 1)$, dont on réduit à nouveau la troncature à l'ordre n (coût $2\, \mathsf{E}_{\mathbb{K}}(n)$). Enfin, on reconstruit les évaluations de $P_1, ..., P_r$ à partir de leurs réductions (coût $r\, \mathsf{E}_{\mathbb{K}}(n)$). Le résultat suit en rajoutant les différents coûts. □

Corollaire. *Si P est quadratique, on peut l'évaluer utilisant $(r + d)\, \mathsf{E}_{\mathbb{K}}(n) + (t - d)\, \mathsf{N}(n)$ opérations dans $\mathbb{K}$.*

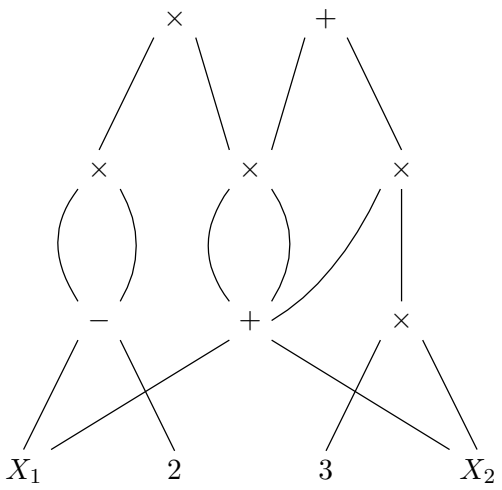


Fig. 4.3. Exemple d'un dag.

4.4.3 Évaluation rapide de polynômes en plusieurs variables

Si les polynômes P sont donnés comme combinaisons linéaires de monômes dans $X_1^{\mathbb{N}} \cdots X_d^{\mathbb{N}}$, alors il faut construire un dag pour P avant d'appliquer la proposition 4.1. Ceci conduit au problème comment trouver un dag pour lequel le nombre de sous-dags multiplicatifs actifs est minimal. Bien sûr, ce problème de nature plutôt combinatoire est assez délicat, donc on cherche plutôt de bonnes stratégies pour s'approcher du minimum.

Soit S l'ensemble des monômes dans $X_1^{\mathbb{N}} \cdots X_d^{\mathbb{N}}$ apparaissant dans P. La question se réduit essentiellement à la nouvelle question comment trouver deux sous-ensembles $V, W \subseteq X_1^{\mathbb{N}} \cdots X_d^{\mathbb{N}}$ tels que $S \subseteq VW$ et tels que le cardinal $|V \cup W|$ soit minimal. Deux approches heuristiques semblent raisonnables :

Partage des coordonnées en deux parties

Pour chaque partie $I \subseteq \{1, ..., d\}$ de fonction caractéristique χ_I, on a une projection naturelle

$$\pi_I \colon X_1^{\alpha_1} \cdots X_d^{\alpha_d} \;\mapsto\; X_1^{\chi_I(1)\alpha_1} \cdots X_d^{\chi_I(d)\alpha_d}$$

La projection $\pi_I(S)$ peut se calculer en temps linéaire. L'idée est maintenant de prendre $V = \pi_I(S)$ et $W = \pi_{\{1,\ldots,n\}\setminus I}(S)$ pour un I qui minimise $|V \cup W|$. Si d est grand, ce calcul peut encore devenir trop cher, auquel cas on peut se restreindre aux parties I de la forme $I = \{1, ..., k\}$ avec $1 \leqslant k < n$.

Décomposition par blocs

Étant donnés des entiers $k_1 \geqslant 1, ..., k_d \geqslant 1$, on peut considérer les projections

$$\begin{aligned} \pi \colon X_1^{\alpha_1} \cdots X_d^{\alpha_d} &\;\mapsto\; X_1^{(\alpha_1 \operatorname{div} k_1)k_1} \cdots X_d^{(\alpha_d \operatorname{rem} k_d)k_d} \\ \varrho \colon X_1^{\alpha_1} \cdots X_d^{\alpha_d} &\;\mapsto\; X_1^{\alpha_1 \operatorname{rem} k_1} \cdots X_d^{\alpha_d \operatorname{rem} k_d} \end{aligned}$$

avec $\pi\,\varrho = \mathrm{Id}$. L'idée est alors de prendre $V = \pi(S)$ et $W = \varrho(S)$ pour des entiers $k_1, ..., k_d$ bien choisis. On peut trouver ces entiers en commençant avec $k_1 = \cdots = k_d = 1$ et en doublant l'un des entiers tant que cela fait chuter la cardinalité de $V \cup W$.

Exemple. Un exemple particulièrement important est quand

$$S = S_{d,k} = \{X_1^{\alpha_1} \cdots X_d^{\alpha_d} : \alpha_1 + \cdots + \alpha_d \leqslant k\},$$

avec

$$S_{d,k} = \binom{k+d}{k}.$$

Pour $d > 1$ la première stratégie donne

$$|V \cup W| \leqslant 2\binom{k + \lceil d/2 \rceil}{k}.$$

Si on peut utiliser la multiplication FFT dans $\mathbb{K}$, la proposition 4.1 implique donc que l'on peut évaluer P en $(f_1, ..., f_d) \in \mathbb{K}[z]/(z^n)$ en temps $\mathcal{O}_{\mathbb{K}}(|S_{d,k}|\, n)$ pour k suffisamment grand. Pour quelques autres résultats intéressants sur la composition de séries formelles en plusieurs variables, nous renvoyons vers [14].

5

Développements certifiés en série

5.1 Algorithmes rapides et numériquement stables

5.1.1 Instabilité numérique de la multiplication rapide

Malheureusement, les algorithmes asymptotiquement rapides pour multiplier des polynômes à coefficients flottants sont numériquement instables dans le cas général. Par exemple, considérons le calcul suivant d'un carré par la méthode de Karatsuba, en utilisant une précision de 4 chiffres binaires :

$$\begin{aligned} P(z) &= 1.0000 + 1.0000 \cdot 2^{-64} z \\ P(z)^2 &= 1.0000^2 + \\ &\quad [(1.0000 + 1.0000 \cdot 2^{-64})^2 - 1.0000^2 - (1.0000 \cdot 2^{-64})^2]\, z + \\ &\quad (1.0000 \cdot 2^{-64})^2 z^2 \\ &= 1.0000 + (1.0000 - 1.0000 - 1.0000 \cdot 2^{-128})\, z + 1.0000 \cdot 2^{-128} z^2 \\ &= 1.0000 - 1.0000 \cdot 2^{-128} z + 1.0000 \cdot 2^{-128} z^2 \end{aligned}$$

On voit bien que le fait d'avoir ajouté et soustrait des coefficients d'ordres de grandeur différents a pollué le calcul, de manière à rendre le coefficient en z du résultat incorrect.

5.1.2 Préconditionnement par mise à l'échelle

Une manière simple pour régler le problème de la section précédente est de rendre tous les coefficients de P de même ordre de grandeur modulo une postcomposition par $\lambda\, z$ pour une « échelle » λ bien choisie :

$$P^2 = [P \circ (2^{64} z)]^2 \circ (2^{-64} z).$$

Malheureusement, cela ne règle pas le problème dans le cas général, car le choix d'un bon λ peut être ambigu. Par exemple, si

$$P = 1.0000 + 1.0000\, z + 1.0000 \cdot 2^{-64} z^2,$$

il faudrait prendre $\lambda = 1$ pour que P_0 et P_1 soient du même ordre de grandeur, mais $\lambda = 2^{64}$ pour que P_1 et P_2 le soient.

Heureusement, dans le cas d'une série $f \in \mathbb{C}[[z]]$, le comportent des f_n est généralement assez régulière pour $n \to \infty$, car dicté par la nature de la singularité de f qui est la plus proche de l'origine. Dans de nombreux cas (et notamment quand f est algébrique avec une unique singularité dominante), on a par exemple

$$|\log|f_n| + n \log \varrho| \;=\; (\log n)^{O(1)},$$

où ϱ est le rayon de convergence de f. Dès lors, l'existence d'une bonne échelle $\lambda = \varrho$ est garantie pour tout polynôme $P = f_n z^n + \cdots + f_{n+\delta} z^{n+\delta}$ avec n et δ suffisamment grands. D'un point de vue pratique, λ s'approche bien, en considérant la pente au milieu du polygone numérique de Newton (voir la section 6.3).

5.1.3 Calcul numériquement stable des nombres de Bell

Afin de vérifier la stabilité numérique de la méthode de préconditionnement par mise à l'échelle, il suffit d'effectuer le même calcul avec des précisions différentes. Bien entendu, cela ne garantit rien, mais ça indique que tout se passe comme prévu.

```
Mmx] use "analyziz"
Mmx] bit_precision := 64;
Mmx] z == series (0.0, 1.0);
Mmx] B == exp (exp z - 1)
```

$1.0000000000000000000 + 1.0000000000000000000\ z + 1.0000000000000000000\ z^2 + 0.83333333333333333369\ z^3 + 0.625000000000000000054\ z^4 + 0.43333333333333333364\ z^5 + 0.28194444444444444471\ z^6 + 0.17400793650793650794\ z^7 + 0.10267857142857142857\ z^8 + 0.058275462962962962678\ z^9 + O(z^{10})$

```
Mmx] B[10000]
```

$5.593910855120672200 85e - 7996$

```
Mmx] bit_precision := 128;
Mmx] z == series (0.0, 1.0);
Mmx] B == exp (exp z - 1);
Mmx] B[10000]
```

$5.59391085512067159074423017436837541267 9e - 7996$

5.2 Certification du calcul des coefficients

5.2.1 Inversion

Pour une séries $g^\bullet$ à coefficients dans $\mathcal{B}(\mathbb{R}, \mathbb{R})$, obtenue comme le résultat d'un calcul naïf comme

$$g^\bullet = \frac{1}{1 - z + z^2},$$

il est instructif d'étudier le comportement des rayons r_n des $g_n^\bullet = \mathcal{B}(c_n, r_n)$ par rapport à celui des centres c_n. La différence principale entre les calculs de r_n et c_n est que « tous les $-$ ont été remplacés par des $+$ » dans les formules de récurrence. En effet, dans notre exemple, on a

$$\begin{aligned} c_n &\approx c_{n-1} - c_{n-2} \\ r_n &\approx r_{n-1} + r_{n-2}. \end{aligned}$$

Par conséquent, si on calcule avec une précision de p chiffres binaires, on obtient

$$r_n \geqslant \frac{\varepsilon}{1 - z - z^2},$$

pour un certain $\varepsilon \approx 2^{-p}$. Comme le rayon de convergence de la série $\varepsilon/(1-z-z^2)$ est plus petite que le rayon de convergence de $1/(1-z+z^2)$, il s'en suit qu'il existe une constante c telle que l'algorithme d'inversion naïf perd toute sa précision pour $n \geqslant c\,p$.

Le remède est pareil que pour l'inversion des matrices dans la section 3.2.5 : supposons que l'on veuille inverser une série $f^\bullet = \mathcal{B}(c, r) \in \mathcal{B}(\mathbb{R}, \mathbb{R})[[z]]$. Alors on calcule un inverse numérique approché u de c, et on prend $1/f^\bullet = (1/(u\,f^\bullet))\,u$, où $1/(u\,f^\bullet)$ est calculé utilisant une méthode détendue générique.

5.2.2 Exponentiation et résolution d'équations différentielles

Considérons maintenant le cas du calcul certifié d'une exponentielle $g^\bullet = \exp f$. Si on fait une analyse similaire à celle de la section précédente, on se rend compte que les séries génératrices c et r formées par les centres et rayons des coefficients $g_n^\bullet = \mathcal{B}(c_n, r_n)$ vérifient cette fois-ci

$$\begin{aligned} c &\approx \exp f \\ r &\leqslant \varepsilon \exp |f|, \qquad \varepsilon \approx 2^{-p}, \end{aligned}$$

où $|f|$ est la série génératrice avec $|f|_n = |f_n|$. Or les rayons de convergence de f et $|f|$ sont les mêmes et les singularités dominantes généralement de même nature. Par magie, on observe donc rarement de la surestimation dans ce cas ! Cette observation s'étend d'ailleurs aux intégrales de systèmes dynamiques plus générales.

Ceci dit, dans les rares cas où on observe quand même de la surestimation, on peut bien sûr adapter la méthode perturbative. En effet, pour le calcul de $\exp f^\bullet$ avec $f^\bullet = \mathcal{B}(c, r)$, il suffit de prendre $\tilde{g} \approx \exp c$, après quoi la fonction $h = \exp f/\tilde{g}$ se calcule en intégrant le système dynamique $h' = (f' - \tilde{g}'/\tilde{g})\,h$.

5.2.3 Calcul certifié des nombres de Bell

```
Mmx] use "analyziz"
Mmx] bit_precision := 64;
Mmx] z == series (0.0, 1.0);
Mmx] B == exp (exp z - 1);
Mmx] B[10000]
```

$5.593910855120672200 85e - 7996$

```
Mmx] z == series (ball 0.0, ball 1.0);
Mmx] B == exp (exp z - 1);
Mmx] B[10000]
```

$5.593910855121e-7996$

5.3 Modèles de Taylor

5.3.1 Définition

Pour l'instant, on a surtout vu des boules scalaires dans $\mathcal{B}(\mathbb{R},\mathbb{R})$ ou $\mathcal{B}(\mathbb{C},\mathbb{R})$ et des boules qui opèrent coefficient par coefficient, comme des boules dans $\mathcal{B}(\mathbb{R}^n,\mathbb{R}^n)$ ou $\mathcal{B}(\mathbb{R}^{n\times n},\mathbb{R}^{n\times n})$. Il est maintenant temps d'étendre les principes du calcul à des espaces un peu plus sophistiqués comme l'ensemble $\mathscr{A}(\mathcal{B}(0,r),\mathbb{R})$ des fonctions réelles analytiques sur la boule $\mathcal{B}(0,r)$. La norme naturelle sur cet espace est donnée par

$$\|f\| \;=\; \sup_{|z|\leqslant r} |f(z)|.$$

Pour $\varphi\in\mathscr{A}(\mathcal{B}(0,r),\mathbb{R})$ et $K\in\mathbb{R}$, on peut donc considérer la boule fermée

$$\mathcal{B}_r(\varphi,K) \;=\; \{f\in\mathscr{A}(\mathcal{B}(0,r),\mathbb{R}):\|f-\varphi\|\leqslant K\}.$$

Étant donné un ordre $n\in\mathbb{N}$, on peut ensuite prendre les centres dans l'ensemble

$$\mathbb{R}[z]_{<n} \;=\; \{P\in\mathbb{R}[z]:\deg P<n\}.$$

Une boule $\mathcal{B}_r(P,K)$ avec $P\in\mathbb{R}[z]_{<n}$ et $K\in\mathbb{R}^{\geqslant}$ est appelé un *modèle de Taylor* d'ordre n sur la boule $\mathcal{B}(0,\,r)$. On note par $\mathcal{B}_r(\mathbb{R}[z]_{<n},\,\mathbb{R})$ l'ensemble des modèles de Taylor de ce type. Pour des calculs sur machine, on peut évidemment définir la variante $\mathcal{B}_r(\mathbb{D}[z]_{<n},\mathbb{D})$.

5.3.2 Opérations

Multiplication

La multiplication doit être définie avec un peu plus de soin pour les modèles de Taylor :

$$\begin{aligned}
&\mathcal{B}_r(f_0+\cdots+f_{n-1}\,z^{n-1},K)\,\mathcal{B}_r(g_0+\cdots+g_{n-1}\,z^{n-1},L)\\
=\;&\mathcal{B}_r(f_0\,g_0+\cdots+(f_0\,g_{n-1}+\cdots+f_{n-1}\,g_0)\,z^{n-1},E^{\mathrm{tr}})\\
E^{\mathrm{tr}}\;=\;&|f_1|\,|g_{n-1}|+|f_2|\,(|g_{n-1}|+|g_{n-2}|)+\cdots+|f_{n-1}|\,(|g_1|+\cdots+|g_{n-1}|).
\end{aligned}$$

Si on calcule avec des flottants de précision p, il faut en plus prendre en compte les erreurs d'arrondi, par exemple en rajoutant

$$E^{\mathrm{rnd}} \;=\; [|f_0|\,|g_{n-1}|+\cdots+(|f_0|+\cdots+|f_{n-1}|)\,|g_0|]\,2^{1-p}$$

au rayon en utilisant systématiquement le mode d'arrondi vers le haut pour les estimations d'erreur. Dans tous les cas, on observe que le calcul des erreurs est négligeable devant le coût de la multiplication polynomiale tronquée pour n pas trop petit.

Intégration

Contrairement au boules traditionnelles, les modèles de Taylor permettent aussi l'implantation d'opérations fonctionnelles comme l'intégration :

$$\int \mathcal{B}_r(f_0 + \cdots + f_{n-1}\, z^{n-1}, K) \;=\; \mathcal{B}_r(f_1 + \cdots + \tfrac{f_{n-2}}{n-1}\, z^{n-1}, K\, r + \tfrac{|f_{n-1}|}{n}\, r^n).$$

Ici, on utilise le fait que $\|\int \varphi\| \leqslant r\, \|\varphi\|$ pour tout $\varphi \in \mathscr{A}(\mathcal{B}(0,r), \mathbb{R})$.

5.3.3 Généralisations

Les définitions se généralisent naturellement aux cas des fonctions analytiques complexes et des fonctions analytiques en d variables sur un polydisque $\mathcal{B}(0,r)$ avec $r \in (\mathbb{R}^{\geqslant})^d$. Les centres d'un modèle de Taylor en plusieurs variables vivent généralement dans un ensemble du type

$$\mathbb{K}[z]_{<\lambda n} \;=\; \{P \in \mathbb{K}[z_1, ..., z_d] \colon \lambda_1\, i_1 + \cdots + \lambda_d\, i_d \geqslant n \Rightarrow P_{i_1,\ldots,i_d} = 0\},$$

avec $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, $n \in \mathbb{N}$ et $\lambda \in (\mathbb{R}^{>})^d$.

5.4 Réduction de la surestimation

5.4.1 L'idée sur un exemple

Une application importante des modèles concerne la réduction de la surestimation quand on évalue une fonction en arithmétique de boules. Revenons à l'exemple 3.3 :

$$\begin{aligned} f^{\bullet}(x^{\bullet}) &= (x^{\bullet})^2 -^{\bullet} 2\, x^{\bullet} +^{\bullet} 1 \\ \chi_{f^{\bullet}}(x) &= \frac{|x|+1}{|x-1|} \end{aligned}$$

Considérons maintenant l'évaluation de l'expression $f(x) = x^2 - 2\, x + 1$ en un modèle de Taylor d'ordre 2 (ou plus) :

$$f^{\mathrm{Taylor}}(\mathcal{B}_r(x,0)) \;=\; \mathcal{B}_r(y_0 + y_1\, t, \varepsilon).$$

Pour tout $u \in \mathcal{B}(x,r)$ on a donc $f(u) \in y_0 + y_1\, (u-x) + \mathcal{B}(0,\varepsilon) \subseteq \mathcal{B}(y_0, |y_1|\, r + \varepsilon)$, ce qui nous permet de définir une meilleure extension $f^{\sharp} \colon \mathcal{B}(\mathbb{R},\mathbb{R}) \to \mathcal{B}(\mathbb{R},\mathbb{R})$ de f par

$$f^{\sharp}(\mathcal{B}(x,r)) \;=\; \mathcal{B}(y_0, |y_1|\, r + \varepsilon)$$

Il est facile de vérifier que cette extension est ponctuellement optimale dans le sens que

$$\chi_{f^{\sharp}}(x) \;=\; 1$$

pour tout x. Bien sûr, il faudra recourir à des modèles de Taylor d'ordre au moins μ si f présente des racines de multiplicité μ (ou lorsqu'une petite perturbation de f présente de telles racines).

5.4.2 Application à la recherche de solutions réelles d'un système polynomial

Considérons un système de polynômes réels

$$P(x) \;=\; 0, \qquad x = (x_1, ..., x_d), P \in \mathbb{R}[x]^d.$$

Un algorithme classique pour trouver les solutions de ce système dans une « boîte » $[0,1]^d$ est de la découper en des boîtes $x^\bullet$ de plus en plus petites jusqu'au moment où l'une des deux situations suivantes se présente :

- On a $0 \notin P(x^\bullet)$, et donc P n'admet pas de racines sur $x^\bullet$.
- On peut démontrer que P admet une unique solution dans $x^\bullet$ (par exemple, en utilisant la méthode de Krawczyk-Rump qui sera exposée dans la section 6.6.1).

Si P n'admet que des racines simples dans $[0, 1]^d$, alors cet algorithme simpliste termine. Toutefois, en présence de racines multiples, ou racines proches (éventuellement dans des voisinages imaginaires), l'évaluation de $P(x^\bullet)$ présentera une surestimation qui rend la méthode inutilisable.

En utilisant les modèles de Taylor, on aimerait donc réduire cette surestimation. Le principal problème est alors de trouver un ordre de troncature adéquat. Ceci peut se faire de façon heuristique. Afin de simplifier l'exposition, considérons le cas de dimension un. Supposons que l'on a calculé un encadrement $P^{<n}(x^\bullet)$ de P sur $x^\bullet$ en utilisant des modèles de Taylor à l'ordre n. Pour un coût $\lambda \in [2, 4]$ plus grand on peut aussi calculer un encadrement $P^{<2n}(x^\bullet)$ en utilisant des modèles de Taylor à l'ordre $2\,n$. Même si on ne connait pas les surestimations $\chi^{<n}$ et $\chi^{<2n}$ de $P^{<n}(x^\bullet)$ et $P^{<2n}(x^\bullet)$dans l'absolu, le quotient entre les rayons de $P^{<n}(x^\bullet)$ et de $P^{<n}(x^\bullet)$ nous fournit le ratio $\chi^{<n}/\chi^{<2n}$. On s'attend donc à ce que l'on doive découper la boîte $x^\bullet$ en des boîtes $\chi^{<n}/\chi^{<2n}$ fois plus petites en utilisant des modèle de Taylor d'ordre n plutôt que $2\,n$. Lorsque $\lambda\,\chi^{<2n} < \chi^{<n}$, on a donc intérêt à prendre des modèles de Taylor d'ordre $2\,n$ plutôt que des modèles d'ordre n. Ce critère nous permet de trouver un ordre à peu près optimal.

Nous notons en outre que les modèles de Taylor en plusieurs variables nous permettent en principe des prendre des boîtes qui ne sont pas forcément parallèles aux axes (voir aussi la section 7.2.3) et donc de mieux suivre la géométrie du problème. En fait, les boîtes ont même le droit de devenir courbées, si cela peut arranger les estimations d'erreur.

5.5 Intégration de systèmes dynamiques

L'intégration certifiée de systèmes dynamiques utilisant la technique des modèles de Taylor a été proposé par Berz et Makino [49, 50]. Un grand avantage de ces méthodes est qu'ils permettent d'étudier la dépendance du flot par rapport aux conditions initiales pour des conditions initiales dans des domaines relativement grands. Bien sûr, ceci nécessite de recourir à des modèles de Taylor en plusieurs variables, c'est à dire par rapport au temps et par rapport aux conditions initiales. Dans cette section, nous allons plutôt nous intéresser à des techniques qui continuent à marcher en haute précision. Dans ce cadre, les conditions initiales sont forcément spécifiées de façon très précise, donc on peut se contenter de modèles de Taylor en une variable. Dans la section 5.5.2, on aura juste besoin d'étudier la dépendance du flot par rapport aux conditions initiales à l'ordre un.

5.5.1 Algorithme de base

Supposons que l'on veuille intégrer le système dynamique

$$\begin{aligned} Y' &= P(Y) \\ Y(z_0) &= C \end{aligned}$$

avec $Y = (Y_1, ..., Y_d)$, $P \in \mathbb{Q}[Y]^d$, $z_0 \in \mathbb{Q}[\mathrm{i}]$ et $C \in \mathcal{B}(\mathbb{D}, \mathbb{D})^d$. Plus précisément, supposons que l'on veuille calculer la valeur $Y(z_1)$ de Y en $z_1 \in \mathbb{Q}[\mathrm{i}]$. Alors on peut utiliser la méthode suivante :

1. Comme le système ne dépend pas du temps, on peut supposer sans perdre de généralité que $z_0 = 0$.
2. Supposant que l'on calcule avec une précision de p chiffres binaires, on sélectionne un ordre n de développement proportionnel à p (disons $n = p$), et on calcule une approximation numérique de la solution autour de $z = 0$:

$$Y \;\approx\; Y_0 + \cdots + Y_{n-1}\, z^{n-1} + \mathcal{O}(z^n).$$

3. Pour $Y^\bullet = \mathcal{B}_{|z_1|}(Y_0 + \cdots + Y_{n-1}\, z^n, K)$, où K est une constante dont le calcul sera détaillé plus bas, on évalue $C + \int P(Y^\bullet)$ utilisant l'arithmétique pour les modèles de Taylor. Si

$$C + \textstyle\int P(Y^\bullet) \;\subseteq\; Y^\bullet, \tag{5.1}$$

   alors on retourne $Y^\bullet(z_1)$.
4. Sinon, on calcule récursivement $Y(z_1/2)$, puis $Y(z_1)$ en fonction de $Y(z_1/2)$.

Vérifions que cette méthode est correcte. En effet, si on a l'inclusion (5.1), alors l'opérateur de Φ: $\psi \mapsto C + \int P(\psi)$ envoie la boule $Y^\bullet$ dans lui même. Comme $Y^\bullet$ est une boule compacte dans un espace de Banach, il s'en suit que Φ admet un point fixe. Or il existe une unique façon de calculer les coefficients en série d'un tel point fixe, donc ce point fixe est nécessairement unique.

Il nous reste à montrer comment calculer la constante K. L'idée est tout simplement de commencer avec $K = 0$ et d'itérer quelque fois le processus suivant : on calcule $\mathcal{B}_{|z_1|}(Y_0 + \cdots + Y_{n-1}\, z^n, K)$ et on remplace K par le supremum de K et le rayon du résultat. Si cette itération diverge numériquement, alors on arrête et (5.1) ne sera pas vérifier. Sinon, on continue jusqu'au moment où K ne bouge plus trop, disons $K := K + \mathrm{d}K$, et on refait une nouvelle fois $K := K + \mathrm{d}K$ avant de finir.

5.5.2 Effet d'enveloppement

Si on applique l'algorithme de la section précédente tel quel dans le cas du système

$$Y' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} Y$$

on observe un effet d'enveloppement similaire au calcul de puissances successives d'un nombre complexe (voir la section 3.2.2, ainsi que la figure 5.1). Pour y remédier, on utilise les idées suivantes [53].

Idée 1 : calculer la dependence en fonction de la condition initiale

$$\begin{aligned} Y(z,\varepsilon) &= C+\varepsilon+\int_0^z P(Y(t,\varepsilon))\,\mathrm{d}t \\ J(z)=\frac{\partial Y}{\partial \varepsilon}(z,0) &= I+\int_0^z \frac{\partial P}{\partial Y}(Y(t,0))\,\frac{\partial Y}{\partial \varepsilon}(t,0)\,\mathrm{d}t. \end{aligned}$$

Idée 2 : encadrements de l'erreur dans des coordonnées déterminées par $\tilde{J} \approx J$

$$\begin{aligned} Y^\bullet(z) &= Y_{\mathrm{cen}}(z)+\tilde{J}(z)\,\mathcal{B}(0,Y_{\mathrm{rad}}(z)) \\ \tilde{J}_{21}\left(\tilde{J}_1\,\mathcal{B}_1+E\right) &= \tilde{J}_{21}\,\tilde{J}_1\left(\mathcal{B}_1+\tilde{J}_1^{-1}\,E\right) \\ &=: \tilde{J}_2\,\mathcal{B}_2. \end{aligned}$$

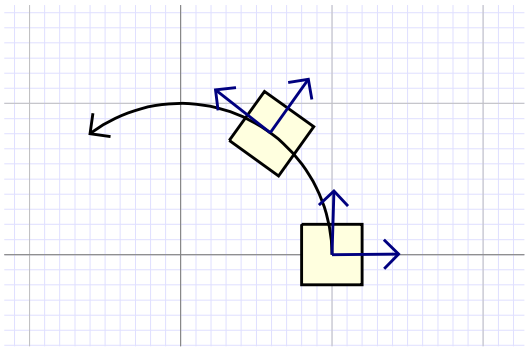


Fig. 5.1. Afin de limiter l'effet d'enveloppement, il faut calculer les estimations d'erreur dans un système de coordonnées qui correspond à la première variation du flot.

5.5.3 Variante pour les modèles de Taylor

Il est naturel de se demander s'il ne faudrait pas remplacer la définition de l'ensemble des modèles de Taylor sur le disque $\mathcal{B}(0,r)$ et d'ordre n par quelque chose d'un peu plus fin, comme

$$\{f_0^\bullet+\cdots+f_{n-1}^\bullet\,z^{n-1}+\mathcal{B}_r(0,K)\,z^n\colon f_0^\bullet,\ldots,f_{n-1}^\bullet\in\mathcal{B}(\mathbb{D},\mathbb{D}),K\in\mathbb{D}\}.$$

Bien sûr, la définition classique est plus simple, ce qui rend les opérations de base plus efficaces. Toutefois, en grande précision, le calcul avec des boules dans $\mathcal{B}(\mathbb{D},\mathbb{D})$ ne coûte pas sensiblement plus cher que le calcul avec des flottants dans $\mathbb{D}$. Par ailleurs, la définition alternative admet aussi quelques avantages :

- Avant tout, les bornes obtenues sont souvent de meilleure qualité. Par exemple, dans le cas de l'intégration, on profite d'une division par $n+1$:

$$\begin{aligned} &\int f_0^\bullet+\cdots+f_{n-1}^\bullet\,z^{n-1}+\mathcal{B}_r(0,K)\,z^n \\ =\;& f_1^\bullet+\cdots+\tfrac{f_{n-2}^\bullet}{n-1}\,z^{n-1}+\mathcal{B}(0,\tfrac{\lceil f_{n-1}^\bullet\rceil}{n}+\tfrac{K}{n+1}) \end{aligned}$$

Comme on le verra dans la section suivante, cette amélioration n'est pas anodine.

- Il n'est pas nécessaire de calculer avec une précision $p \geqslant c\,n$ pour avoir des bornes précises pour le reste $f_n\,z^n + f_{n+1}\,z^{n+1} + \cdots$.
- Les bornes obtenues donnent des bornes plus fines quand on restreint le domaine $\mathcal{B}(0,r)$ du modèle de Taylor à une boule $\mathcal{B}(0,s)$ plus petite.

5.5.4 Qualité des estimations

Examinons plus en détails la qualité des estimations lorsque l'on utilise la variante des modèles de Taylor de la section précédente. On suppose que l'on a calculé des encadrements $f_0^\bullet, ..., f_{n-1}^\bullet$ pour les coefficients de la solution $f_0, ..., f_{n-1}$ jusqu'à l'ordre n et on pose

$$\varphi^\bullet \;=\; f_{;n}^\bullet = f_0^\bullet + \cdots + f_{n-1}^\bullet\, z^{n-1}.$$

Notre but est d'obtenir un encadrement $\varepsilon^\bullet$ pour le reste

$$\varepsilon \;=\; f_n + f_{n+1}\, z + \cdots$$

Soit Φ la fonctionnelle définie par

$$\Phi(\psi) \;=\; C + \int P(\psi).$$

En calculant $\Phi(\varphi^\bullet)$ en utilisant l'arithmétique pour les modèles de Taylor, on trouve une boule $\delta^\bullet = \mathcal{B}_r(0,\delta)$ avec

$$\Phi(\varphi^\bullet) \;\subseteq\; \varphi^\bullet + \delta^\bullet\, z^n.$$

Rappelons que notre but était de trouver un encadrement $\varepsilon^\bullet$ pour ε avec

$$\Phi(\varphi^\bullet + \varepsilon^\bullet\, z^n) \;\subseteq\; \varphi^\bullet + \varepsilon^\bullet\, z^n.$$

Or

$$\begin{aligned}\Phi(\varphi^\bullet + \varepsilon^\bullet\, z^n) \;&=\; \Phi(\varphi^\bullet) + J_\Phi(\varphi^\bullet)\,\varepsilon^\bullet\, z^n + \mathcal{O}((\varepsilon^\bullet\, z^n)^2)\\ &\subseteq\; \varphi^\bullet + \delta^\bullet\, z^n + \tfrac{r}{n}\, J_P(\varphi^\bullet)\,\varepsilon^\bullet\, z^n.\end{aligned}$$

Nous en sommes donc réduit à trouver un $\varepsilon^\bullet$ avec

$$\delta^\bullet + \tfrac{r}{n}\, J_P(\varphi^\bullet)\,\varepsilon^\bullet \;\subseteq\; \varepsilon^\bullet.$$

Supposant que

$$\|J_P(\varphi^\bullet)\| \;\leqslant\; \frac{n}{r}, \tag{5.2}$$

il suffit alors de prendre

$$\varepsilon^\bullet \;=\; \frac{\delta^\bullet}{1 - \left\|\tfrac{r}{n}\, J_P(\varphi^\bullet)\right\|}. \tag{5.3}$$

L'intérêt de majorer le reste $f_n\,z^n + f_{n+1}\,z^{n+1} + \cdots$ par une expression de la forme $\mathcal{B}_r(0,\varepsilon)\,z^n$ et non pas par une boule de la forme $\mathcal{B}_r(0,\varepsilon)$ saute désormais au yeux : dans le deuxième cas, il aurait fallu demander que

$$\|J_P(\varphi^\bullet)\| \;\leqslant\; \frac{1}{r},$$

et on aurait dû se résoudre à prendre

$$\varepsilon^\bullet \;=\; \frac{\delta^\bullet}{1 - \|r\,J_P(\varphi^\bullet)\|}.$$

La taille de nos pas aurait donc été environ n fois trop pessimiste.

La formule (5.3) montre aussi qu'il est intéressant de rendre $\|J_P(\varphi^\bullet)\|$ aussi petit que possible. Ceci suggère de ne pas chercher des majorations directes dans les coordonnées d'origine, mais plutôt des majorations de la forme $\varepsilon^\bullet = T\,\eta^\bullet$ tel que $\|T^{-1}\,J_P(\varphi^\bullet)\,T\|$ soit petit.

On observe enfin que la condition (5.2) est vérifiée en prenant n suffisamment grand. À partir de cette observation, on montre que le rayon r pour lequel on peut certifier l'existence d'une borne tend vers le rayon de convergence r_f de f lorsque l'on fait tendre p et n vers l'infini. Dans la terminologie du chapitre 2, cela implique $r_f \in \mathbb{R}^{\mathrm{calg}}$

5.5.5 Exemple dans Mathemagix

L'équation de Volterra est souvent utilisée comme benchmark pour des systèmes d'intégration certifiés. L'implantation des modèles de Taylor est en cours dans Mathemagix et dans la session qui suit nous trichons encore un peu quant à la « certification ». Toutefois, l'exemple montre bien ce que nous cherchons à faire un jour.

```
Mmx] use "continewz"
Mmx] type_mode? := false;
Mmx] x: Coordinate == coordinate ('x);
Mmx] y: Coordinate == coordinate ('y);
Mmx] sys: Vector MVPolynomial Rational == [ 2 * x * (1 - y), - y * (1 - x) ]
```

$[-2\,x\,y + 2\,x, x\,y - y]$

```
Mmx] coords: Vector Coordinate == [ x, y ]
```

$[x, y]$

```
Mmx] init == [ as_double 1.0, as_double 3.0 ]
```

$[1.000, 3.000]$

```
Mmx] ode_order := 48;
Mmx] ode_precision := 48;
Mmx] significant_digits := 4;
```

```
Mmx] solve_ode_taylor (sys, coords, init)
```

$$[1.000 - 4.000\,z + \cdots + 1.298e10\,z^{46} - 9.244e10\,z^{47} + B(7.578e-15, 0.250),\\ 3.000 - 6.000\,z^2 + \cdots + 4.662e10\,z^{47} + B(3.413e-15, 0.250)]$$

```
Mmx] first_variation_taylor (sys, coords, init)
```

$$\begin{bmatrix} 1.000 - 4.000\,z + \cdots + B(9.085e-15, 0.2500) & -2.000\,z + 8.000\,z^2 - \cdots + B(7.459e-15, 0.2500) \\ 3.000\,z - 6.000\,z^2 - \cdots + B(5.225e-15, 0.2500) & 1.000 - 5.000\,z^2 + \cdots + B(3.603e-15, 0.2500) \end{bmatrix}$$

```
Mmx]
Mmx] significant_digits := 0;
Mmx] period == as_double 5.48813846815;
Mmx] [ @integrate_ode_taylor (sys, coords, init, period * (t / 15)) ||
       t in 0 to 15 ]
```

$$\begin{bmatrix} 1.000000000000 & 3.0000000000000 \\ 0.2639512719212 & 2.532704326551 \\ 0.110182566396 & 1.868766969917 \\ 0.0715291625253 & 1.337897339600 \\ 0.064867569728 & 0.950826195310 \\ 0.074788854601 & 0.67623406024 \\ 0.102116743645 & 0.48415042304 \\ 0.156823946955 & 0.3517090769739 \\ 0.260948734324 & 0.2627872752886 \\ 0.457593143500 & 0.207086884012 \\ 0.826954421344 & 0.180415632864 \\ 1.50562753694 & 0.189426934661 \\ 2.65951143189 & 0.277013241481 \\ 4.05454863893 & 0.66289616945 \\ 3.3945484972 & 2.0166233447 \\ 1.0000000000 & 3.000000000000 \end{bmatrix}$$

```
Mmx]
Mmx] integrate_ode_taylor (sys, coords, init, period)
```

$[1.0000000000, 3.000000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 2 * period)
```

$[0.9999999999, 3.00000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 5 * period)
```

$[1.000000000, 3.00000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 10 * period)
```

$[1.00000000, 3.00000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 20 * period)
```

$[1.00000000, 3.0000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 50 * period)
```

$[1.0000000, 3.0000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 100 * period)
```

$[1.0000000, 3.0000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 200 * period)
```

$[1.000000, 3.000000000]$

```
Mmx] integrate_ode_taylor (sys, coords, init, 500 * period)
```

$[1.00000, 3.000000000]$

6

Calcul des racines d'un polynôme en une variable

D'une part, il existe des algorithmes d'une bonne complexité théorique pour calculer les racines d'un polynôme en une variable [66, 59, 60]. D'autre part, il existe des bons implantations [35, 7]. La messe est-elle dite ? Malheureusement, l'état de l'art présente encore plusieurs défauts.

D'une part, les algorithmes asymptotiquement rapides marchent bien quand la précision de calcul p excède le degré du polynôme. Or une précision bien moindre suffit généralement pour isoler la plupart des racines. Par ailleurs, il est souvent plus rapide d'appliquer des algorithmes numériques brutaux en précision machine, plutôt que de payer un grand facteur constant pour l'utilisation d'une arithmétique « rapide » en précision multiple. D'autant plus que ces algorithmes brutaux sont généralement plus faciles à parallèliser. Manifestement le passage des algorithmes naïfs mais asymptotiquement mauvais à des algorithmes intelligents mais asymptotiquement bons pose encore un problème dans ce domaine.

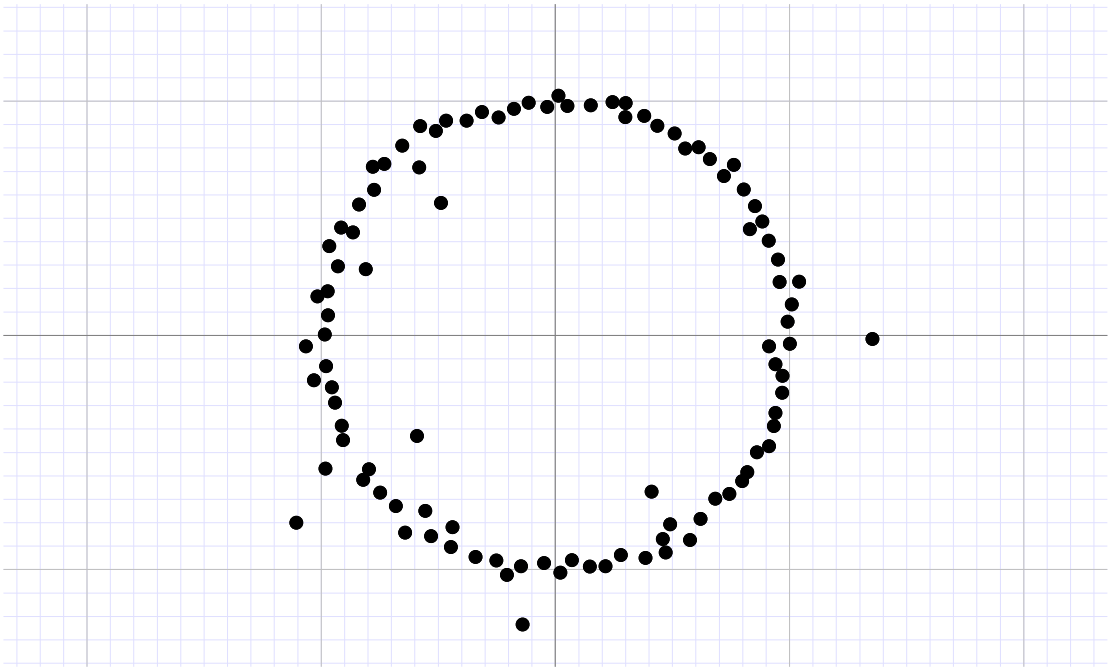
En outre, trouver *toutes* les racines dans le *cas général* pose de sérieux problèmes techniques : souvent les bonnes idées ne s'appliquent que dans certains cas et il est difficile de les combiner. D'ailleurs, les algorithmes de Pan attendent toujours à être implantés.

Enfin, il est fort probable que les mesures de complexité traditionnels ne soient pas tout à fait opérationnels pour ce problème. En effet, il existe de nombreux types de polynômes (voir la section 6.1) et l'efficacité et la qualité des solveurs dépend grandement de la répartition géométrique des racines. Faudrait-il des nombres de conditionnement plus fins pour exprimer la vraie complexité ? Est-ce qu'il vaudrait mieux considérer un autre problème, comme la factorisation en deux facteurs de degrés deux fois moindre ? Je ne connais pas les réponses pour l'instant, d'où le caractère plus « expérimental » de ce chapitre.

6.1 Exemples pour différents types de polynômes

6.1.1 Polynômes tirés au hasard

```
Mmx] use "analyziz";
Mmx] include "graphix/points.mmx";
Mmx] pi == 4.0 * arctan 1.0;
Mmx] rnd () ==
       exp (complex (0.0, uniform_deviate (0.0, 2*pi)));
Mmx] p == polynomial (rnd () | i in 0 to 100);
Mmx] $points roots p
```

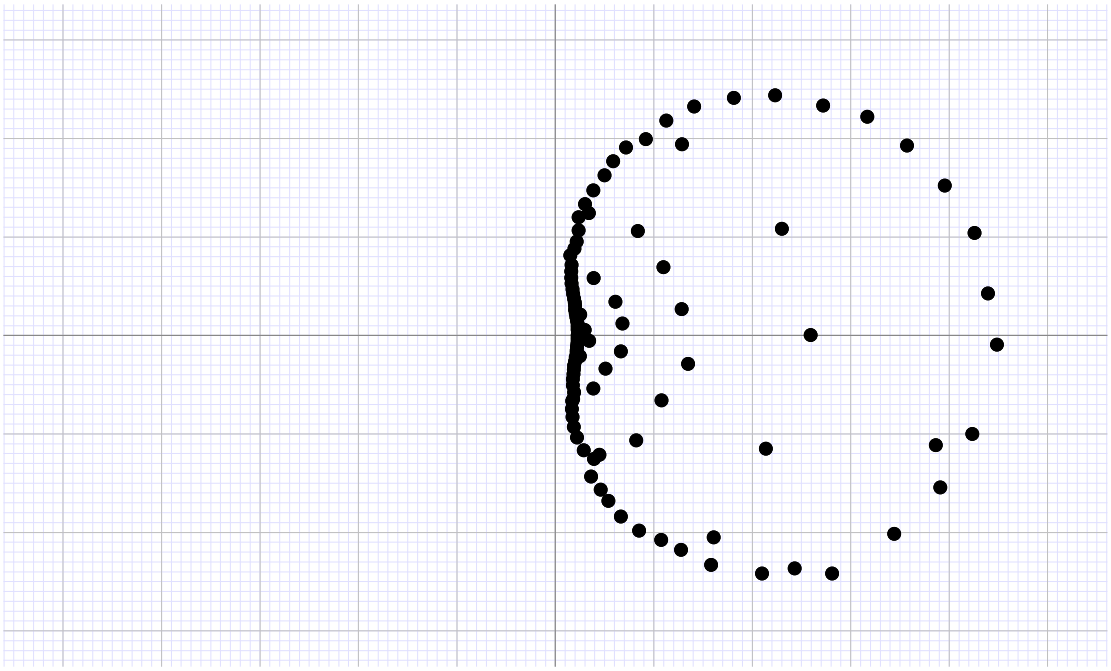


Meilleur conditionnement :

$$\begin{aligned} P(z) &= z^n - 1 + E(z) \\ z_k &= \mathrm{e}^{2\pi \mathrm{i} k/n} + O(n\,\|E\|) \\ \|E\| &= \sup_{|z|\leqslant 1} \|E(z)\| \end{aligned}$$

6.1.2 Polynômes avec zéros de grandes multiplicités

```
Mmx] p == poly (1.0, -1.0)^100;
Mmx] $points (roots p)
```



Pire conditionnement :

$$\begin{aligned} P(z) &= (z-1)^n + E(z) \\ z_k &= 1 + O\left(\sqrt[n]{\|E\|}\right) \end{aligned}$$

Pour différentes multiplicités :

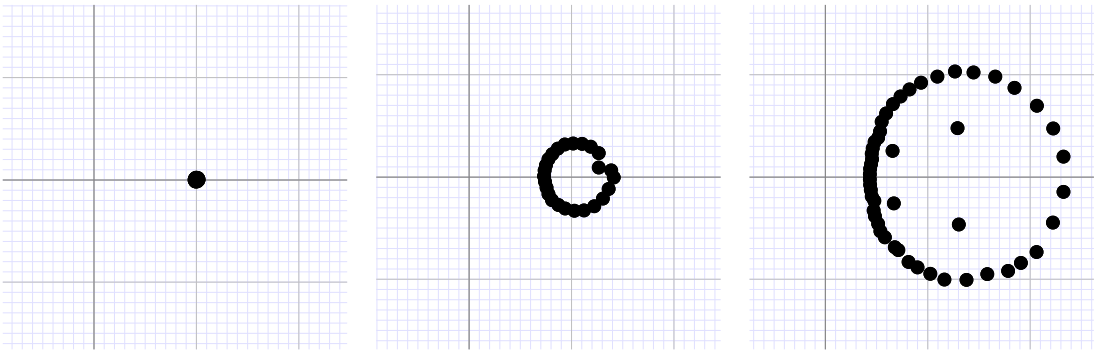
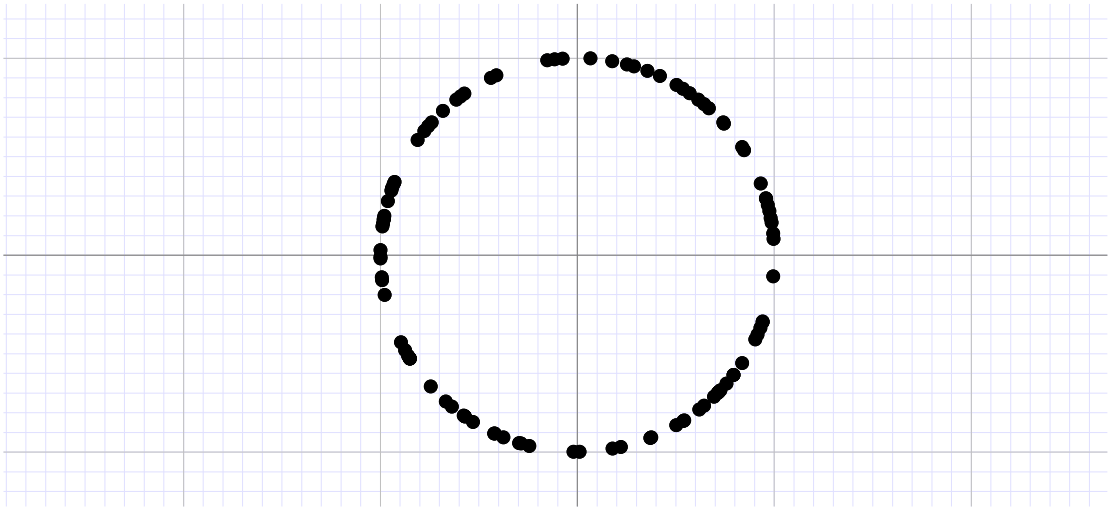


Fig. 6.1. Racines de $(z-1)^k$ pour $k = 10, 25, 50$ et en calculant avec une précision de 64 bits.

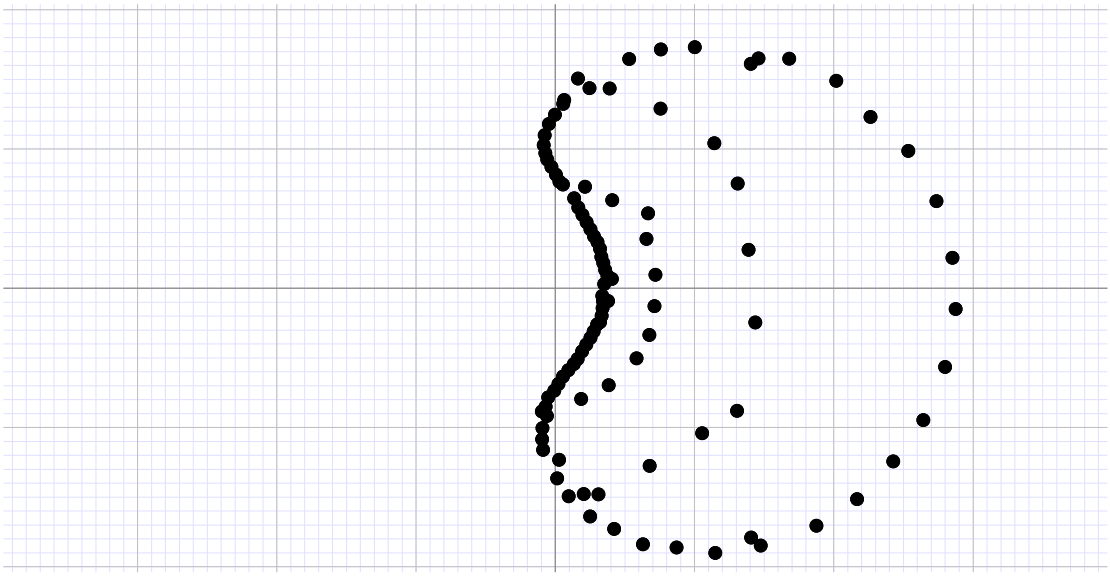
6.1.3 Expérience à propos du conditionnement

On pourrait penser que le mauvais conditionnement est lié à la présence d'une racine de grande multiplicité (modulo petites perturbations du moins). L'exemple qui suit montre qu'il n'en est rien : des polynômes dont toutes les racines sont similaires en norme et dans le même demi plan sont à peu près aussi mal conditionnés qu'un polynôme de la forme $(z-\sigma)^n$.

```
Mmx] v == [ rnd () | i in 1 to 100 ];
Mmx] p == annulator v;
Mmx] $points (roots p)
```

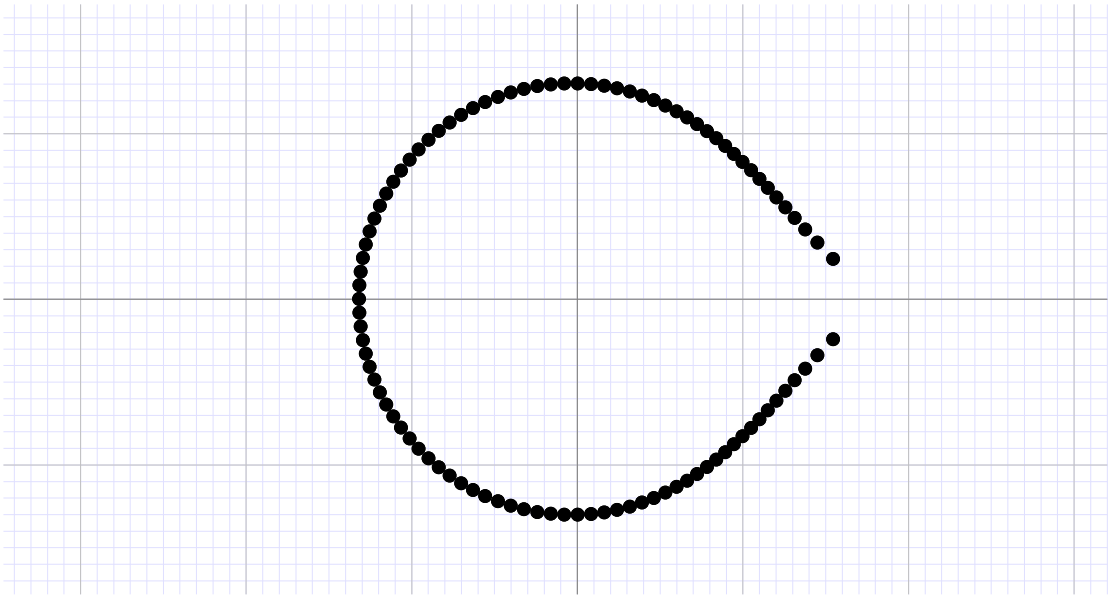


```
Mmx] v == [ sqrt rnd () | i in 1 to 100 ];
Mmx] p == annulator v;
Mmx] $points (roots p)
```

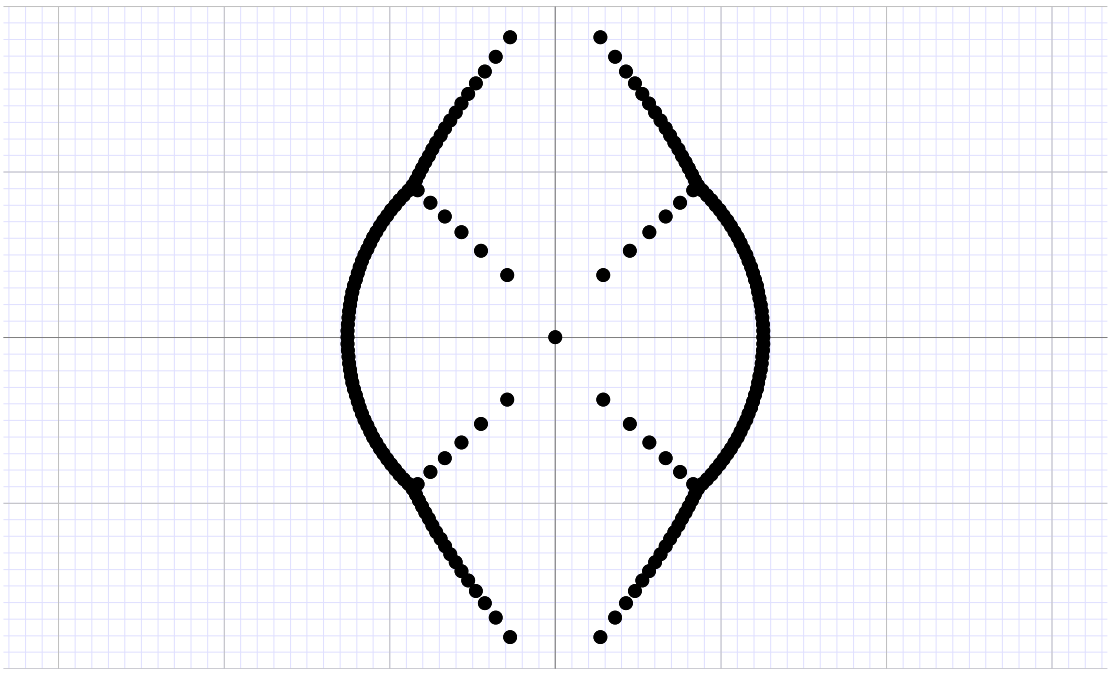


6.1.4 Séries formelles tronquées

```
Mmx] z == series (0.0, 1.0);
Mmx] B == exp (exp z - 1);
Mmx] p == B[0,100];
Mmx] $points (0.5 * roots p)
```



```
Mmx] z == series (0.0, 1.0);
Mmx] f == integrate exp (-z*z);
Mmx] p == f[0,200];
Mmx] $points (0.2 * roots p)
```



6.2 Quelques méthodes numériques classiques

6.2.1 Itération d'Aberth

Un algorithme numérique classique et efficace pour trouver les racines d'un polynôme

$$P = P_n z^n + \cdots + P_0 \in \mathbb{C}[z]$$

est la méthode d'Aberth. On part d'un ansatz $z_1, ..., z_n \in \mathbb{C}$ pour les racines, comme

$$z_k \;=\; \mathrm{e}^{2\pi \mathrm{i} k/n}.$$

L'idée est ensuite d'appliquer pour chaque k la méthode de Newton à la fonction

$$f(z) \;=\; \frac{P(z)}{\prod_{j \neq k} (z - z_j)},$$

ce qui conduit à l'itération

$$z'_k \;:=\; z_k - \frac{P(z_k)}{P'(z_k) - P(z_k) \sum_{j \neq k} \frac{1}{z_k - z_j}}.$$

Une propriété intéressante de cet algorithme est le fait que dès que l'on a trouvé des bonnes approximations de $l < k$ racines de P, disons $z_1, ..., z_l$, alors les itérations suivantes appliquent essentiellement la méthode d'Aberth pour le polynôme

$$Q(z) \;=\; \frac{P(z)}{(z - z_1) \cdots (z - z_l)}$$

et les racines restantes z_{l+1}, ..., z_k. La méthode est particulièrement efficace quand on l'applique de façon naïve en précision machine (avec une complexité de calcul en $\mathcal{O}(n^2)$) ; c'est une des raisons de l'efficacité du logiciel MPSOLVE [7].

6.2.2 Méthode par homotopie

La méthode par homotopie est un autre exemple d'une méthode qui est efficace quand on l'applique de façon naïve en précision machine. L'itération est un peu différente que pour la méthode d'Aberth, mais son efficacité est du même ordre de grandeur. On y reviendra dans un contexte plus général dans le chapitre 7. Voir aussi [71].

6.2.3 Transformation de Graeffe

La plupart des algorithmes asymptotiquement efficaces font intervenir la transformation de Graeffe $P \mapsto G(P)$, dont voici la définition :

$$\begin{aligned} P(z) &= P_{\mathrm{pair}}(z^2) + P_{\mathrm{imp}}(z^2)\, z \\ G(P)(z) &= P_{\mathrm{pair}}(z)^2 - P_{\mathrm{imp}}(z)^2\, z \end{aligned}$$

On vérifie que

$$\begin{aligned} P_{\text{pair}}(z^2) &= \tfrac{1}{2}\,(P(z)+P(-z)) \\ P_{\text{imp}}(z^2) &= \tfrac{1}{2\,z}\,(P(z)-P(-z)) \\ G(P)(z^2) &= \tfrac{1}{4}((P(z)+P(-z))^2-(P(z)-P(-z))^2) \\ &= P(z)\,P(-z) \end{aligned}$$

Il s'en suit la propriété fondamentale que $\deg P = \deg G(P)$ et

$$P(z)=0 \;\Rightarrow\; G(P)(z^2)=0.$$

En d'autres termes, la transformation est un séparateur puissant de racines, au moins en norme : si $|z_1|\,(1+\varepsilon)<|z_2|$ pour deux racines z_1 et z_2 de P, alors $|z_1^2|\,(1+2\,\varepsilon)<|z_2^2|$ pour les racines z_1^2 et z_2^2 de $G(P)$. Or, lorsque les racines $z_1,\ldots,z_n$ de P sont très différents en module, disons $|z_1| \ll \cdots \ll |z_n|$, alors on a $z_k \approx P_{k-1}/P_k$ pour chaque $k \in \{1,\ldots,n\}$. Si les racines $z_1,\ldots,z_n$ de P sont toutes différentes en module, il suffit donc de quelques transformations de Graeffe pour obtenir $z_1^{2^k},\ldots,z_n^{2^k}$ pour un certain k.

Reste le problème comment retrouver $z_1,\ldots,z_n$ à partir de $z_1^{2^k},\ldots,z_n^{2^k}$. On peut utiliser une astuce pour cela qui consiste à calculer avec des « nombres tangents » dans $\mathbb{C}[\epsilon]/(\epsilon^2)$ au lieu de simples coefficients dans $\mathbb{C}$. En effet, lorsque l'on fait le remplacement

$$P(z) \;\rightsquigarrow\; P(z-\epsilon)=P(z)-P'(z)\,\epsilon,$$

et que l'on calcule les puissances 2^k-ièmes des racines de $P(z-\epsilon)$ en faisant quelques transformations de Graeffe, on obtient pour chaque racine $z+\epsilon$:

$$\begin{aligned} (z+\epsilon)^{2^k} &= z^{2^k}+2^k\,z^{2^k-1}\,\epsilon = u+v\,\epsilon \\ z &= \frac{u}{2^k\,v} \end{aligned}$$

6.3 Polygone numérique de Newton

Nous venons de voir l'importance de l'étude des normes des racines du polynôme P au delà de leur calcul à proprement parler. Le polygone numérique de Newton nous fournit les valeurs approximatives de ces normes, directement à partir des coefficients du polynôme.

Le polygone de Newton intervient aussi lorsque l'on veut développer une arithmétique efficace *et* numériquement stable pour les polynômes. Une telle arithmétique reste typiquement précis pour le calcul de P^2 avec $P=1+z+1.0000\cdot 2^{-100}\,z^2$. Un tel algorithme de multiplication est donné dans [85], mais la borne (3) dans ce papier est fausse.

6.3.1 Exemple d'un polygone numérique de Newton

Le polygone numérique de Newton de $P = P_n\,z^n + \cdots + P_0$ est défini comme l'enveloppe convexe de l'ensemble

$$\{(i, y-\log|P_i|)\colon i\in\{0,\ldots,n\},\, y\geqslant 0\}.$$

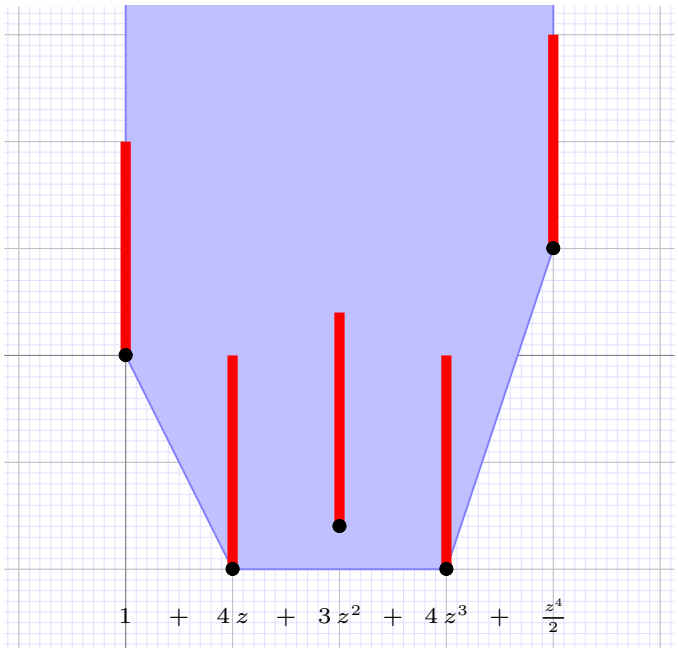


Fig. 6.2. Exemple d'un polygone numérique de Newton

6.3.2 Évaluation du polynôme et polygone numérique de Newton

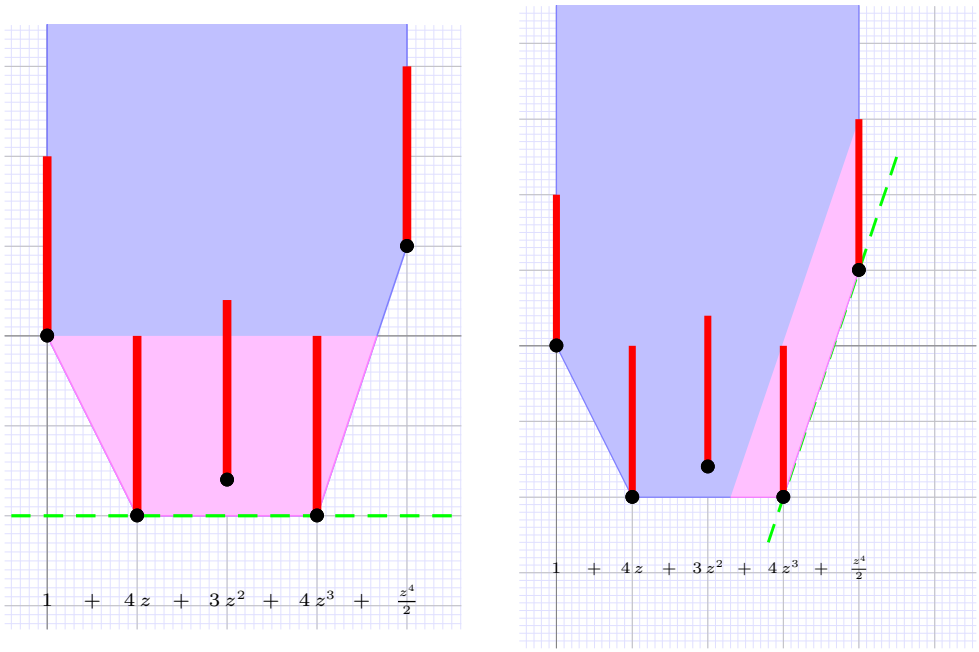


Fig. 6.3. Quand on évalue le polynôme en un point z avec $|z| = 1$ (à gauche) ou $|z| = 8$, seul la zone mauve des coefficients de P contribue significativement au résultat (lorsque l'on calcule avec deux bits de précision).

Moralité : Polygone de Newton $\rightsquigarrow$ Comportement en évaluation

En particulier : Pentes du polygone de Newton $\rightsquigarrow$ Modules des racines

$$\begin{array}{lcll} z_1 & \approx & -0.289 & |z_1| \approx 1/4 \\ z_2 & \approx & -0.193 - 0.952\,\mathrm{i} & |z_2| \approx 1 \\ z_3 & \approx & -0.193 + 0.952\,\mathrm{i} & |z_3| \approx 1 \\ z_4 & \approx & -7.325 & |z_4| \approx 8 \end{array}$$

La question suivante semble ouverte et abordable :

Question. *Supposons des pentes $\alpha_1 \leqslant \cdots \leqslant \alpha_n$ et des racines : $|z_1| \leqslant \cdots \leqslant |z_n|$.*
Est-ce qu'il existe une constante $c \geqslant 1$ universelle avec :

$$|\log_2 |z_k| - \alpha_k| \leqslant c \log n \ ?$$

6.4 Stabilité numérique et efficacité : un mariage difficile

Problème. Meilleure approximation possible des racines de P pour précision p donnée.

Note. Sans perdre en généralité, on peut perturber les coefficients de P par des nombres d'erreur relative $\leqslant 2^{-p}$. Ceci peut par exemple être utile si on veut éviter que toutes les racines soient de même module.

L'espoir

- Arithmétique naïve en précision machine
  Algorithme en temps $C\,n^2$ avec C petit ?
- Arithmétique rapide en précision multiple
  Nécessite une précision d'au moins $p \geqslant n$
  Algorithme en temps $C\,\mathsf{I}((p+n)\,n) \log^{O(1)} n$, avec C pas trop grand ?
- Compromis lorsque $p = \mathcal{O}(n)$?

Plus réaliste

Pour un certain facteur $K_{\kappa,n}$ dépendant du conditionnement κ et de n :

- Arithmétique naïve en précision machine
  Algorithme en temps $C\,K_{\kappa,n}\,n^2$ avec C petit ?
- Arithmétique rapide en précision multiple
  Algorithme en temps $C\,K_{\kappa,n}\,\mathsf{I}((p+n)\,n) \log^{\mathcal{O}(1)} n$, avec C pas trop grand ?
- Compromis lorsque $p = \mathcal{O}(n)$?

Exemple d'un théorème précis :

Théorème 6.1. (Schönhage [66]) *Pour $P \in \mathbb{Z}[z]\,2^{-p}$ avec $\|P\| \leqslant 1$, on peut calculer en temps $\mathcal{O}(\mathsf{I}((p + n\log n)\,n)\,n)$ des nombres $a_k, b_k \in \mathbb{Z}\,2^{-p}$ avec*

$$\|P - (a_1 + b_1\,z) \cdots (a_n + b_n\,z)\| \ \leqslant \ 2^{-p}.$$

Remarque. Gourdon a implanté la méthode de Schönhage [35]. D'autres résultats de complexité ont été donné par Pan [59, 60].

6.5 Évaluation multi-point rapide

C'est le problème inverse de la recherche des racines. En effet, ce problème est important lorsque l'on veut raffiner une approximation pour une précision p en une meilleure approximation pour la précision $2\,p$ (par exemple, en utilisant la méthode de Newton).

Arithmétique naïve en précision machine

Coût $\mathsf{E}(n,p) = \mathcal{O}(\mathsf{I}(p)\,n^2)$, en appliquant n fois Horner.

Arithmétique rapide en précision multiple

$\mathsf{E}(n,p) = \mathcal{O}(\mathsf{I}((p+n)\,n)\log n)$

Algorithme intermédiaire

$\mathsf{E}(n,\,p) = \mathcal{O}(\mathsf{I}(n^{3/2}\,p^{3/2}\log^3 n))$, en faisant des FFT massives sur différents cercles de centre 0 [84]. Cet algorithme est meilleur que les précédents lorsque $p = o(\sqrt[3]{n})$.

Question. *Peut-on faire mieux lorsque* $p = \mathcal{O}(n)$ *?*

6.6 Certification des racines

Les algorithmes de Schönhage et de Pan, qui admettent les meilleures complexités actuelles, certifient déjà les résultats. Toutefois, en suivant la même philosophie que dans les autres chapitres, il peut être utile de découpler les étapes de calcul et la certification des résultats.

6.6.1 Racines simples

Certification par la méthode de Krawczyk [43], présentée pour le cas de fonctions en plusieurs variables :

$$\begin{aligned} f\colon \mathbb{C}^n \to \mathbb{C}^n, \qquad & \mathcal{B}(z,r) \in \mathcal{B}(\mathbb{C}^n, \mathbb{R}^n), \qquad V \approx J_f(z)^{-1} \\ K(\mathcal{B}(z,r)) \;&=\; z - Vf(z) + (1 - VJ_f(\mathcal{B}(z,r)))\,\mathcal{B}(0,r) \end{aligned}$$

Théorème 6.2. *Si* $K(\mathcal{B}(z,r)) \subseteq \mathcal{B}(z,r)$*, alors* $f(u) = 0$ *pour un certain* $u \in \mathcal{B}(z,r)$*.*

Démonstration. Soit $g(z) = z - Vf(z)$. Pour $u \in \mathcal{B}(z,r)$, on a

$$\begin{aligned} g(u) \;&=\; g(z) + \int_0^1 J_g(z + (u-z)\,t)\,(u-z)\,\mathrm{d}t \\ &\subseteq\; g(z) + J_g(\mathcal{B}(z,r))\,\mathcal{B}(0,r), \end{aligned}$$

puisque $J_g(\mathcal{B}(z,r))$ est convexe. Donc

$$g(\mathcal{B}(z,r)) \;\subseteq\; g(z) + J_g(\mathcal{B}(z,r))\,\mathcal{B}(0,r) \;\subseteq\; \mathcal{B}(z,r),$$

et g admet un point fixe, grace à la compacité de $\mathcal{B}(z,r)$. □

Ce théorème ne garantit pas l'unicité de la racine. Rump [61, 63] a donné une version plus fine du théorème qui garantit aussi l'unicité :

Théorème 6.3. *Si $K(\mathcal{B}(z,r)) \subseteq \operatorname{int}\mathcal{B}(z,r)$, alors $f(u)=0$ pour un unique $u \in \mathcal{B}(z,r)$.*

Pour trouver r, faire

$$\begin{aligned} r_0 &= 0 \\ r_{k+1} &= (1+\varepsilon)\max(r_k, \operatorname{rad} K(z,r_k)) \end{aligned}$$

$\rightsquigarrow$ Méthode de ε-inflation de Rump [61, 63].

6.6.2 Racines multiples

Problème. Soit P un polynôme avec un comportement

$$P(z) \;\approx\; (z-\sigma)^\mu + P_{\mu+1}(z-\sigma)^{\mu+1} + \cdots + P_n(z-\sigma)^n$$

pour $z \in \mathcal{B}(\sigma,r)$. Comment montrer que P admet exactement μ racines sur $\mathcal{B}(\sigma,r)$?

Approche par modèles de Taylor

Prendre $\boldsymbol{u} = \mathcal{B}_r(\sigma+z,0) \in \mathcal{B}_r(\mathbb{C}[z]_{\leqslant\nu},\mathbb{R})$ pour $\nu \geqslant \mu$ et évaluer

$$\boldsymbol{v} \;=\; P(\boldsymbol{u}) = \mathcal{B}_r(Q_0 + \cdots + Q_\nu z^\nu, \varepsilon).$$

Montrer que tout $\tilde{P} \in \boldsymbol{v}$ admet μ racines dans $\mathcal{B}(0,r)$.
Par exemple, il suffit de vérifier que

$$\varepsilon + |Q_0| + \cdots + |Q_{\mu-1}|\,r^{\mu-1} + |Q_{\mu-1}|\,r^{\mu+1} + \cdots + |Q_\nu|\,r^\nu < |Q_\mu|\,r^\mu$$

et d'appliquer le théorème de Rouché. Voir [87] pour des variantes de cette méthode.

Approche par déflation

Une autre approche consiste à examiner de combien il faut déformer le polynôme $P \rightsquigarrow \tilde{P} = P + E$ pour qu'il admette *exactement* une racine de multiplicité μ. Cette approche « par déflation » [63] permet de se ramener à la théorie de la section 6.6.1 en cherchant à résoudre simultanément les équations $\tilde{P}(z) = \cdots = \tilde{P}^{(\mu-1)}(z) = 0$. Toutefois, l'astuce introduit un facteur $\mathcal{O}(\mu^3)$ dans la complexité en temps.

Remarque 6.4. Voir aussi la section 7.3 pour un point de vue différent sur la question.

7

Méthodes par homotopie

7.1 Introduction

7.1.1 Historique

Dans la section 1.3.1, nous avons esquissé comment résoudre des systèmes polynomiaux par homotopie. Cette technique ancienne remonte à la démonstration par Gauss que $\mathbb{C}$ est algébriquement clos. En relation avec des implantations concrètes, les méthodes par homotopie apparaissent dans plusieurs contextes différents mais liés :

Homotopies numériques

Les homotopies numériques non certifiés ont été étudié par de nombreux auteurs. Voir par exemple [23, 55, 90, 73] et les références dans ces documents. Quelques logiciels connus pour des homotopies numériques sont PHCpack et Bertini [91, 4].

Complexité théorique des méthodes par homotopie

La complexité théorique des méthodes par homotopie a été analysé dans [71, 72, 69, 70, 21, 5]. Ces travaux donnent une justification théorique pourquoi les méthodes par homotopie marchent aussi vite, tout en faisant des hypothèses assez fortes sur le modèle de calcul.

Homotopies algébriques

Au lieu de calculer avec des nombres flottants, on peut aussi définir les trajectoires de manière algébrique. Ce point de vue a donné lieu à la méthode de Kronecker [34, 33, 24], qui admet également une implantation concrète [46]. Par construction, cette méthode est algébrique donc certifiée, bien qu'elle ne tire pas profit de l'efficacité du calcul numérique pur. Par ailleurs, ce point de vue évite les chemins partant vers l'infini ; voir la section 7.1.2 plus bas.

Homotopies numériques certifiées

Il est naturel de chercher à certifier les méthodes numériques par homotopie, mais curieusement, peu d'efforts ont été déployés dans ce sens. Quelques exceptions : [42, 47, 88]. Des implantations partielles sont présentes dans Mathemagix et Macaulay 2 depuis 2009.

7.1.2 Première difficulté : solutions partant vers l'infini

Les méthodes par homotopie ne viennent pas sans leur lot de problèmes. D'une part, si le système est surdéterminé, le problème est mal posé d'un point de vue numérique. Dans ce cas, le problème est intrinsèquement irrésoluble par des voies exclusivement numériques, du moins si on souhaite certifier les solutions. Or, même en excluant des systèmes surdéterminés, et en considérant exclusivement des systèmes zéro dimensionnels avec n équations et n inconnus, les méthodes par homotopie comportent plusieurs inconvénients.

Premièrement, il est crucial que le système initial « facile à résoudre » ressemble autant que possible au système final qui nous intéresse réellement. En particulier, en prenant un système du même multi-degré, on espère que les deux systèmes admettent le même nombre de solutions. Malheureusement, tel n'est pas toujours le cas, comme on le voit sur l'exemple

$$\begin{aligned} P_1 &= x^2 - 2\,x \\ P_2 &= x\,y - 2 \end{aligned}$$

qui admet seulement une solution $x = 2 \wedge y = 1$ au lieu de $4 = 2 \times 2 = \deg P_1 \deg P_2$. Dans ce cas, trois des quatre chemins de l'homotopie partent vers l'infini. On peut remédier au problème des chemins qui partent vers l'infini en homogénéisant le système et en coupant par un hyperplan affine générique :

$$\begin{aligned} P_1 &= x^2 - 2\,x\,z \\ P_2 &= x\,y - 2\,z^2 \\ P_3 &= 3\,x - 5\,y + 7\,z - 10 \end{aligned}$$

Pour ce nouveau système, il n'y a plus de chemins qui partent à l'infini, mais il existe toujours trois chemin qui tendent vers une racine multiple qui nous intéresse pas.

Dans ce cas précis, il aurait été possible d'éviter ces trois chemins parasites, en prenant des systèmes initiaux et finaux qui ne se ressemblent pas seulement en degré mais aussi quant à leurs polytopes de Newton. Pour plus de détails, voir par exemple [90] et des travaux plus récents du même auteur. En général, il est difficile de trouver un système initial à résoudre qui ressemble suffisamment au système final pour que chaque solution du système initial se déforme en une unique solution du système final.

7.1.3 Deuxième difficulté : solutions multiples

Une deuxième difficulté concerne les systèmes dégénérés avec des racines multiples. Reprenons l'exemple $P_1(x, y, z) = P_2(x, y, z) = P_3(x, y, z) = 0$ de la section précédente dans MATHEMAGIX :

```
Mmx] use "continewz"
Mmx] include "continewz/homotopy_solve.mmx"
Mmx] type_mode? := false;
```

```
Mmx] time_mode? := false;
Mmx] significant_digits := 4;
Mmx] x == coordinate ('x);
Mmx] y == coordinate ('y);
Mmx] z == coordinate ('z);
Mmx] p == x*x - 2*x*z;
Mmx] q == x*y - 2*z*z;
Mmx] r == 3*x - 5*y + 7*z - 10;
Mmx] homotopy_solve ([p,q,r], [x,y,z], 1.0)
```

$$\begin{bmatrix} 2.500 & 1.250 & 1.250 \\ 9.812e-15-1.699e-14\,\mathrm{i} & -2.000+1.698e-7\,\mathrm{i} & 7.004e-8+1.213e-7\,\mathrm{i} \\ -9.097e-6-0.001020\,\mathrm{i} & -2.032-0.03402\,\mathrm{i} & -0.02253-0.02386\,\mathrm{i} \\ -1.373e-11-2.031e-17\,\mathrm{i} & -2.000+6.599e-13\,\mathrm{i} & 3.706e-6+4.714e-13\,\mathrm{i} \end{bmatrix}$$

Pour une précision fixée, l'algorithme peine à converger vers la solution triple $x = z = 0$, $y = -2$, et s'arrête bien avant d'avoir atteint une bonne qualité d'approximation. Ceci tient au fait que les méthodes naïves de suivi de chemin sont à convergence quadratique pour des solutions simples, mais seulement à convergence linéaire pour des solutions multiples. Une technique pour remédier à ce problème sera discutée dans la section 7.3.

7.2 Algorithmes de suivi de chemin

7.2.1 Cas purement numérique

Méthode « prédicteur-correcteur »

$$\begin{aligned} H(z,t) &\approx 0 \\ H(z+\mathrm{d}z, t+\mathrm{d}t) &\approx 0 \\ \mathrm{d}z &= -\frac{\partial H}{\partial z}(z,t)^{-1}\left(H(z,t)+\frac{\partial H}{\partial t}(z,t)\,\mathrm{d}t\right) && \text{(prédiction)} \\ \mathrm{d}z &\mathrel{+}= -\frac{\partial H}{\partial z}(z,t)^{-1}\,H(z+\mathrm{d}z, t+\mathrm{d}t) && \text{(correction)} \end{aligned}$$

Contrôle de pas

$$\begin{aligned} \mathrm{d}t &:= \lambda\,\mathrm{d}t \quad (\lambda > 1), \text{ si pas accepté} \\ \mathrm{d}t &:= \alpha\,\mathrm{d}t, \text{ sinon} \end{aligned}$$

Critère numérique d'acceptation

$$\left\| \frac{\partial H}{\partial z}(z,t)^{-1}\,\frac{\partial H}{\partial z}(z+\mathrm{d}z, t+\mathrm{d}t) - 1 \right\| \leqslant \text{seuil}$$

Contrôle de la précision

Surestimation $\leqslant \varepsilon\, 2^p$ pour un certain seuil $\varepsilon \ll 1$.

7.2.2 Certification naïve par Krawczyk-Rump uniforme

Toute méthode de certification d'une racine simple d'un système polynomial donne lieu à une méthode de certification pour des algorithmes de suivi de chemin en remplaçant les scalaires par des petite boules qui bornent la dépendance en t de façon uniforme. On peut par exemple utiliser la méthode de Krawczyk-Rump de la section 6.6.1 ou [42]. Géométriquement parlant, ceci revient à certifier que, pour des t dans une certaine boule, le chemin z_t à suivre reste dans une certaine boîte comme dans la figure 7.1.

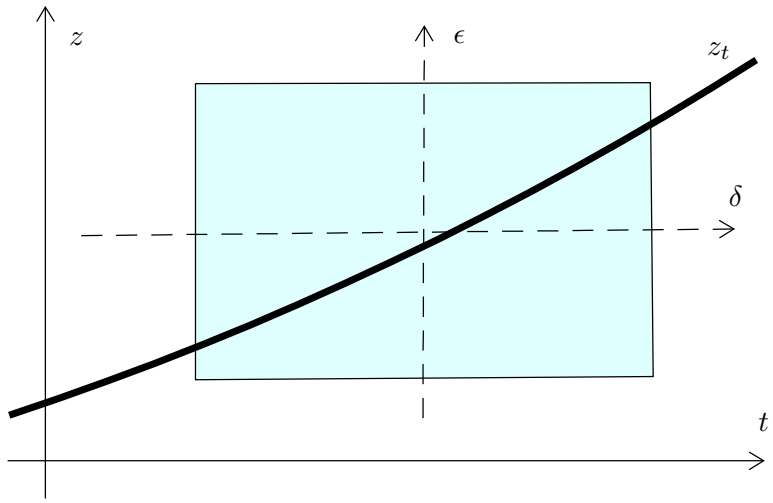


Fig. 7.1. Encadrement du chemin par la méthode de Krawczyk-Rump uniforme.

Remarque 7.1. Afin de certifier les zéros d'un système, il n'est pas forcément nécessaire de certifier les algorithmes de suivi de chemin : si l'algorithme numérique de suivi de chemin produit un nombre suffisant de solutions distinctes, on peut directement utiliser la méthode de Krawczyk-Rump pour les certifier.

7.2.3 Certification plus précise par modèles de Taylor

La méthode de la section précédente conduit à des tailles de pas dt inutilement pessimistes. En effet à cause de la forme géométrique des encadrements, on ne souffre pas seulement de la surestimation de l'erreur due à la variation de t, mais aussi de la surestimation en z. Afin de réduire la surestimation en z, il vaut mieux encadrer le chemin par un ensemble qui suit mieux la courbe. Dans [88, section 6.3], je montre comment faire cela en évaluant l'homotopies H non pas en des boules, mais en des modèles de Taylor d'un type particulier.

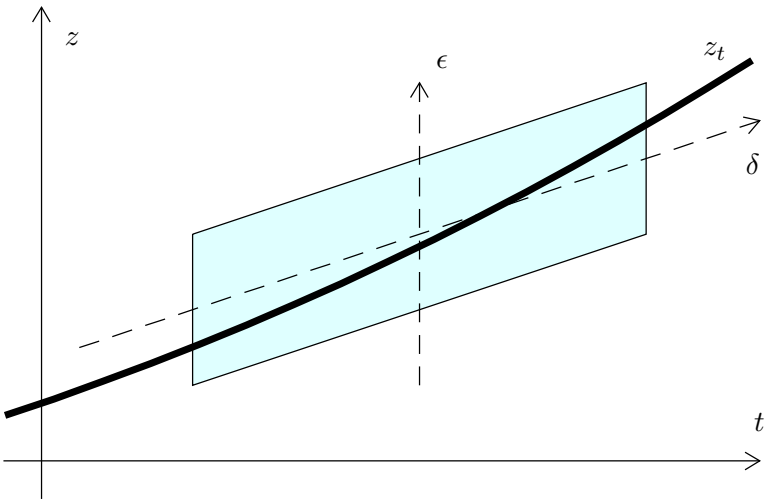


Fig. 7.2. Encadrement du chemin par la méthode de Krawczyk-Rump curviligne d'ordre 1.

7.3 Racines multiples et suivi de troupeaux de solutions

Considérons une homotopie générale

$$H(z,t) \;=\; 0,$$

avec $z=(z_1,...,z_n)\in\mathbb{C}^n$ et $H\in\mathbb{Q}[z_1,...,z_n,t]$. Pour $t\to 0$, toute solution z_t est donnée par une série de Laurent algébrique

$$z_t \;\in\; \mathbb{C}^n((t^{1/\kappa})).$$

En particulier, en remplaçant t par $e^{2\pi ij/\kappa}\,t$ avec $j=\{0,...,\kappa-1\}$, on obtient un « troupeau » de κ solutions. Une méthode [88, section 7.2] pour approcher des solutions multiples consiste à considerer tout le troupeau comme un unique chemin dans un nouvel espace. Cette méthode admet à la fois les avantages d'une déflation [48, 51] et des éclatements (des chemins distincts modulo $\mathcal{O}(z)$ et de même multiplicité κ ont des limites distinctes dans le nouvel espace). En utilisant la FFT, les troupeaux se suivent également bien d'un point de vue numérique [88, section 5.2].

7.3.1 La méthode sur un exemple

$$\begin{aligned} P(z,t) &= 0 \\ \mathbb{A}_{\alpha,\beta} &= \mathbb{C}[u]/((u-\alpha)\,(u-\beta)) \\ P(u,t) &\in \mathbb{A}_{\alpha,\beta} \\ P(u,t)=0 &\;\Leftrightarrow\; P(\alpha,t)=0\wedge P(\beta,t)=0 \end{aligned}$$

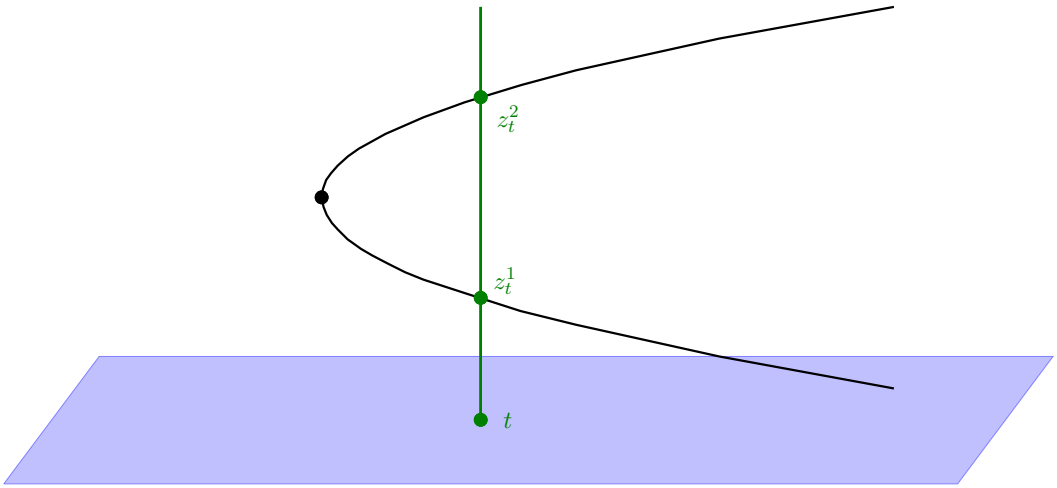


Fig. 7.3. Illustration du suivi simultané d'un troupeau de deux chemins conjugués.

7.3.2 Le cas général

Idée : Considérer $\mathcal{Z}_t = \{z_t^1, ..., z_t^\kappa\}$ comme « chemin » avec idéal $\mathfrak{I}_t$

Représentation de $\mathfrak{I}_t$ (en position générale)

$$\begin{cases} (z_{t,1})^\kappa - U_{t,1}(z_{t,1}) = (z_{t,1} - z_{t,1}^1) \cdots (z_{t,1} - z_{t,1}^\kappa) \\ z_{t,2} - U_{t,2}(z_{t,1}) \\ \vdots \\ z_{t,n} - U_{t,n}(z_{t,1}) \end{cases}$$

Relever l'homotopie H

Évaluer H en $(u, U_{t,2}(u), ..., U_{t,n}, t) \in \mathbb{A}^n \times \mathbb{C}$ avec $\mathbb{A} = \mathbb{C}[u]/(u^r - U_{t,1}(u))$

7.4 Comparaisons avec d'autres méthodes

7.4.1 Un exemple dans Mathemagix

Une question naturelle est de savoir quelle est l'efficacité des méthodes par homotopie par rapport à des algorithmes rapides pour calculer des bases de Gröbner [16, 32, 28]. La session Mathemagix en dessous est éclairante à cet égard. Sur un problème plus ou moins générique (et de ce fait favorisant les méthodes par homotopie) avec 48 solutions, on observe que le *résultat* du calcul des solutions certifiées prend environ une page, alors que la base de Gröbner en prend plus de 6 (pour des raisons d'espace, nous avons tronqué les résultats ici). Plus qu'il y a de solutions, plus que ce ratio entre les tailles tend à devenir grand.

```
Mmx] use "continewz"
Mmx] include "continewz/homotopy_solve.mmx"
Mmx] type_mode? := false;
Mmx] time_mode? := true;
Mmx] bit_precision := 128;
Mmx] significant_digits := 5;
Mmx] x == coordinate ('x);
Mmx] y == coordinate ('y);
Mmx] z == coordinate ('z);
Mmx] p == 5*x*x*x*x - x*x*x*y + x*y*y*x - z*z*y + x*x - 3*x + y + 5;
Mmx] q == 8*y*y*y*y + 3*y*y*z*z - x*y*z - 2*z*z*z + 23*x - y + 11;
Mmx] r == 9*z*z*z - 3*x*x*x - x*y*z - x*y*x + y*y - x + z + 80;
Mmx] homotopy_solve ([p,q,r], [x,y,z], ball 1.0)
```

$$\begin{bmatrix} 0.86665 - 0.78355\,\mathrm{i} & 0.96832 - 0.99058\,\mathrm{i} & 1.0957 - 1.8289\,\mathrm{i} \\ 0.90477 + 0.87836\,\mathrm{i} & 1.2043 - 0.75386\,\mathrm{i} & 0.98365 - 1.8400\,\mathrm{i} \\ -0.75650 + 0.89077\,\mathrm{i} & 1.0641 - 0.29335\,\mathrm{i} & 0.97682 - 1.8069\,\mathrm{i} \\ & \vdots & \\ 0.58140 + 0.69575\,\mathrm{i} & -0.82070 - 1.3655\,\mathrm{i} & -2.0528 + 0.016605\,\mathrm{i} \\ -0.96537 + 0.72340\,\mathrm{i} & -0.45155 - 1.4129\,\mathrm{i} & -2.0756 + 0.019674\,\mathrm{i} \\ -0.85359 - 1.0055\,\mathrm{i} & 0.48345 - 0.99158\,\mathrm{i} & 0.99445 - 1.7608\,\mathrm{i} \end{bmatrix}$$

```
Computed in 406 ms
Mmx] set_ordering graded_reverse_lex_ordering ()
Mmx] total_basis ([p,q,r])          // affichage pas encore dans le bon ordre
```

$$\left[-3\,z^3 + \frac{1}{3}\,x\,y\,z - \frac{1}{3}\,z - \frac{1}{3}\,y^2 + \cdots + \frac{7973}{24}\,x + \frac{27721}{408},\right.$$
$$\frac{639806475}{31296392}\,z^6 + \frac{4392879}{3912049}\,y\,z^5 + \cdots + \frac{5122682256793}{38306783808}\,x + \frac{43873835534195}{27133971864},$$
$$\frac{39916575}{3912049}\,z^6 - \frac{771264}{3912049}\,y\,z^5 - \frac{3939480}{3912049}\,x\,z^5 + \cdots + \frac{265713786517}{4788347976}\,x + \frac{8091385720036}{10175239449},$$
$$\frac{-7036560}{3912049}\,z^6 - \frac{289224}{3912049}\,y\,z^5 - \frac{1477305}{3912049}\,x\,z^5 + \cdots + \frac{10437755597}{598543497}\,x - \frac{1400047676609}{10175239449},$$
$$\frac{11310183375}{1408258711}\,z^7 - \frac{5433034527}{1408258711}\,y\,z^6 - \frac{60585861840}{1408258711}\,x\,z^6 +$$
$$\frac{10802347656839576507}{99165187477959102}\,z^6 + \cdots + \frac{18743638298826335609671317 71}{22285035587246828339692 8},$$
$$\frac{-9428193000}{1408258711}\,z^7 - \frac{2594042712}{1408258711}\,y\,z^6 + \frac{9647300160}{1408258711}\,x\,z^6 + \cdots +$$
$$\frac{457484558173949110036531}{910336421047664556360}\,x - \frac{173674854211762821008 2385}{309514383156205949162 4},$$
$$\frac{-4147675275}{1408258711}\,z^7 + \frac{567798363}{1408258711}\,y\,z^6 + \cdots +$$
$$\left.\frac{628370425983266251703 4103}{963885622285762471440 0}\,x - \frac{736893561967613758957 6891}{409651389471449050362 0}\right],$$

$$\frac{136777417569}{52395701720}\,z^8 + y\,z^7 - \frac{8537435739163780591637}{4980602477214968597100}\,z^7 + \cdots +$$
$$\frac{1166927707431622209724692698750119666847}{139086538637303761985161072946707200 0}\,x -$$
$$\frac{418029078764764453721940234963071727389 71}{118223557841708197687386912004701120 00},$$
$$\frac{-13493777397}{32747313575}\,z^8 + x\,z^7 - \frac{23752500753751269659 38}{62257530965187107463 75}\,z^7 + \cdots +$$
$$\frac{5648970933837839889091052315615755329 7}{4925981576737841570307788000195880000}\,x^2 -$$
$$\frac{8228462179155082103691115885467573141 1}{8692908664831485124072567059169200 00}\,x -$$
$$\frac{2825942213368143079603087286206772063 77}{7388972365106762355461682000293820000},$$
$$z^9 + \frac{23379421200969926407570957}{17910354520100345624491500 0}\,z^8 -$$
$$\frac{5415791150701245285656652052530735684 1}{1021507717186196220951936037728488250 00}\,z^7 + \cdots +$$
$$\frac{1298248565668748543482201849770945925438250279966912599}{7131565569862964704832247407087375537464977156000000}\,x -$$
$$\frac{1255725488924024571134607260902705570967631645913488353}{2424732293753407999642964118409707682738092233040000 00}\Bigg]$$

7.4.2 Discussion et perspectives

Bien sûr, la comparaison que nous venons de faire est favorable aux méthodes par homotopie. Toutefois, elle met en lumière un problème de fond concernant les bases de Gröbner : au moins sur certains exemples, la taille du résultat est beaucoup plus grande que nécessaire.

En outre, si on souhaite avoir une solution numérique, le simple calcul d'une base de Gröbner n'est pas suffisant. Même si on dispose n'une base pour l'ordre lexicographique, il faudra encore trouver les racines d'un gros polynôme en une variable. Or ce dernier problème pourrait se révéler plus dur que le problème de départ. En effet, il y a moins d'espace pour les racines dans $\mathbb{C}$ que dans $\mathbb{C}^n$, ce qui nous impose de calculer avec une plus grande précision p, comme on l'avait déjà constaté dans le chapitre 6. Par contraste, la précision machine suffit dans de nombreux cas, lorsque l'on applique des méthodes par homotopie sur le système de départ.

Toutefois, les systèmes surdéterminés et les chemins partant vers l'infini forment deux problèmes importants pour les méthodes par homotopie. Il est donc fort probable qu'il n'existe pas une technique unique idéale pour la résolution de systèmes polynomiaux.

Ce qui soulève les questions de pouvoir combiner plusieurs méthodes, et, plus modestement, de rendre les résultats d'une méthode utilisables pour d'autres méthodes. À l'intérieur des systèmes de calcul de bases de Gröbner, on dispose par exemple de l'algorithme FGLM pour passer d'un ordre sur les monômes à un autre [29]. De la même manière, on pourra vouloir faire des conversions entre bases de Gröbner, représentations de Kronecker, systèmes triangulaires et des représentations numériques certifiées des variétés de solutions.

Par exemple, lorsque l'on dispose des racines $\alpha_1, ..., \alpha_d$ approchées d'un polynôme unitaire $P \in \mathbb{Q}[z]$, il est facile de reconstituer le polynôme en calculant $P = (z - \alpha_1) \cdots (z - \alpha_d)$ et en retrouvant les coefficients par développement en fractions continues. Cette approche se généralise aux polynômes en plusieurs variables, et permet le calcul d'une représentation de Kronecker à partir de l'ensemble des solutions d'un système zéro-dimensionnel [88, section 7.4]. Ceci permet en particulier l'utilisation de méthodes numériques par homotopie pour la résolution de systèmes surdéterminés.

Références

1 O. Aberth. *Computable analysis*. McGraw-Hill, New York, 1980.

2 G. Alefeld and J. Herzberger. *Introduction to interval analysis*. Academic Press, New York, 1983.

3 ANSI/IEEE. IEEE standard for binary floating-point arithmetic. Technical report, ANSI/IEEE, New York, 2008. ANSI-IEEE Standard 754-2008. Revision of IEEE 754-1985, approved on June 12, 2008 by IEEE Standards Board.

4 D. Bates, J. Hauenstein, A. Sommese, and C. Wampler. Bertini: Software for numerical algebraic geometry. `http://www.nd.edu/~sommese/bertini/`, 2006.

5 C. Beltrán and L.M. Pardo. Smale's 17th problem: Average polynomial time to compute affine and projective solutions. *Journal of the AMS*, 2008. To appear.

6 D. Bini and V.Y. Pan. *Polynomial and matrix computations. Vol. 1*. Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms.

7 D. A. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, 23:127–173, 2000.

8 E. Bishop and D. Bridges. *Foundations of constructive analysis*. Die Grundlehren der mathematische Wissenschaften. Springer, Berlin, 1985.

9 J. Blanck, V. Brattka, and P. Hertling, editors. *Computability and complexity in analysis*, volume 2064 of *Lect. Notes in Comp. Sc.*, Berlin, Heidelberg, 2001. Springer.

10 A. Bostan, F. Chyzak, F. Ollivier, B. Salvy, É. Schost, and A. Sedoglavic. Fast computation of power series solutions of systems of differential equations. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1012–1021, New Orleans, Louisiana, U.S.A., January 2007.

11 A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *Proceedings of ISSAC 2003*, pages 37–44. ACM, 2003.

12 B.L.J. Braaksma. Multisummability and Stokes multipliers of linear meromorphic differential equations. *J. Diff. Eq*, 92:45–75, 1991.

13 R.P. Brent and H.T. Kung. $O((n \log n)^{3/2})$ algorithms for composition and reversion of power series. In J.F. Traub, editor, *Analytic Computational Complexity*, Pittsburg, 1975. Proc. of a symposium on analytic computational complexity held by Carnegie-Mellon University.

14 R.P. Brent and H.T. Kung. Fast algorithms for composition and reversion of multivariate power series. In *Proc. Conf. Th. Comp. Sc.*, pages 149–158, Waterloo, Ontario, Canada, August 1977. University of Waterloo.

15 R.P. Brent and H.T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

16 B. Buchberger. *Ein Algorithmus zum auffinden der Basiselemente des Restklassenringes nach einem null-dimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

17 D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

18 D.V. Chudnovsky and G.V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory (computers in mathematics, stanford, ca, 1986). In *Lect. Notes in Pure and Applied Math.*, volume 125, pages 109–232, New-York, 1990. Dekker.
19 J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.
20 D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proc. of the* 19^{th} *Annual Symposium on Theory of Computing*, pages 1–6, New York City, may 25–27 1987.
21 Jean-Pierre Dedieu. *Points fixes, zéros et la méthode de Newton*. Number 54 in Mathématiques et Applications. SMAI-Springer Verlag.
22 J. Denef and L. Lipshitz. Decision problems for differential equations. *The Journ. of Symb. Logic*, 54(3):941–950, 1989.
23 F.J. Drexler. Eine Methode zur Berechnung sämtlicher Lösungen von Polynomgleichungssystemen. *Numer. Math.*, 29(1):45–58, 1977.
24 C. Durvye. *Algorithmes pour la décomposition primaire des idéaux polynomiaux de dimension nulle donnés en évaluation*. PhD thesis, Univ. de Versailles (France), 2008.
25 J. Écalle. L'accélération des fonctions résurgentes (survol). Unpublished manuscript, 1987.
26 J. Écalle. *Introduction aux fonctions analysables et preuve constructive de la conjecture de Dulac*. Hermann, collection: Actualités mathématiques, 1992.
27 J. Écalle. Six lectures on transseries, analysable functions and the constructive proof of Dulac's conjecture. In D. Schlomiuk, editor, *Bifurcations and periodic orbits of vector fields*, pages 75–184. Kluwer, 1993.
28 J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.
29 J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
30 M. Fürer. Faster integer multiplication. In *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, pages 57–66, San Diego, California, 2007.
31 J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2-nd edition, 2002.
32 A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso. "one sugar cube, please" or selection strategies in the buchberger algorithm. In S. Watt, editor, *Proc. ISSACâĂŹ91*, pages 49–54, New York, 1991. ACM Press.
33 M. Giusti, K. Hägele, J. Heintz, J.E. Morais, J.L Monta na, and L.M. Pardo. Lower bounds for diophantine approximation. *Journal of Pure and Applied Algebra*, 117–118:277–317, 1997.
34 M. Giusti, J. Heintz, J.E. Morais, and L.M. Pardo. When polynomial equation systems can be solved fast? In G. Cohen, M. Giusti, and T. Mora, editors, *Proc. AAECC'11*, volume 948 of *Lecture Notes in Computer Science*. Springer Verlag, 1995.
35 X. Gourdon. *Combinatoire, algorithmique et géométrie des polynômes*. PhD thesis, École polytechnique, 1996.
36 A. Grzegorczyk. Computable functionals. *Fund. Math.*, 42:168–202, 1955.
37 A. Grzegorczyk. On the definition of computable functionals. *Fund. Math.*, 42:232–239, 1955.
38 A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.
39 G. Hanrot, V. Lefèvre, K. Ryde, and P. Zimmermann. MPFR, a C library for multiple-precision floating-point computations with exact rounding. `http://www.mpfr.org`, 2000.
40 L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis*. Springer, London, 2001.
41 A. Karatsuba and J. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.
42 R.B. Kearfott. An interval step control for continuation methods. *SIAM J. Numer. Anal.*, 31(3):892–914, 1994.

43 R. Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehler-schranken. *Computing*, 4:187–201, 1969.
44 V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Springer, 1997.
45 B. Lambov. Interval arithmetic using sse-2. In Peter Hertling, Christoph Hoffmann, Wolfram Luther, and Nathalie Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, volume 5045 of *Lecture Notes in Computer Science*, pages 102–113. Springer Berlin/Heidelberg, 2008.
46 G. Lecerf. *Une alternative aux méthodes de réécriture pour la résolution des systmes algébriques*. PhD thesis, École polytechnique, 2001.
47 A. Leykin. NAG. `http://www.math.uic.edu/~leykin/NAG4M2`, 2009. Macaulay 2 package for numerical algebraic geometry.
48 Anton Leykin, Jan Verschelde, and Ailing Zhao. *Algorithms in algebraic geometry*, chapter Higher-order deflation for polynomial systems with isolated singular solutions, pages 79–97. Springer, New York, 2008.
49 K. Makino and M. Berz. Remainder differential algebras and their applications. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational differentiation: techniques, applications and tools*, pages 63–74, SIAM, Philadelphia, 1996.
50 K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based validated integrators. Technical Report MSU Report MSUHEP 40910, Michigan State University, 2004.
51 A. Mantzaflaris and B. Mourrain. Deflation and certified isolation of singular zeros of polynomial systems. In A. Leykin, editor, *Proc. ISSAC'11*, pages 249–256, San Jose, CA, US, Jun 2011. ACM New York.
52 J. Martinet and J.-P. Ramis. Elementary acceleration and multisummability. *Ann. Inst. Henri Poincaré*, 54(4):331–401, 1991.
53 R.E. Moore. Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions to ordinary differential equation. In L.B. Rall, editor, *Error in Digital Computation*, volume 2, pages 103–140. John Wiley, 1965.
54 R.E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, N.J., 1966.
55 A.P. Morgan. *Solving polynomial systems using continuation for] engineering and scientific problems*. Prentice-Hall, Englewood Cliffs, N.J., 1987.
56 Jean-Michel Muller. *Elementary Functions, Algorithms and Implementation*. Birkhauser Boston, Inc., 2006.
57 A. Neumaier. *Interval methods for systems of equations*. Cambridge university press, Cambridge, 1990.
58 V. Pan. *How to multiply matrices faster*, volume 179 of *Lect. Notes in Math.* Springer, 1984.
59 V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Comput. Math. Appl.*, 31(12):97–138, 1996.
60 V. Y. Pan. Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. *J. Symbolic Comput.*, 33(5):701–733, 2002.
61 S.M. Rump. *Kleine Fehlerschranken bei Matrixproblemen*. PhD thesis, Universität Karlsruhe, 1980.
62 S.M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999.
63 S.M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.
64 L. Schlesinger. *Handbuch der Theorie der linearen Differentialgleichungen*, volume I. Teubner, Leipzig, 1895.
65 L. Schlesinger. *Handbuch der Theorie der linearen Differentialgleichungen*, volume II. Teubner, Leipzig, 1897.
66 A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Math. Inst. Univ. of Tübingen, 1982.
67 A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

68 Alexandre Sedoglavic. *Méthodes seminumériques en algèbre différentielle ; applications à l'étude des propriétés structurelles de systèmes différentiels algébriques en automatique*. PhD thesis, École polytechnique, 2001.
69 M. Shub and S. Smale. Computational complexity. On the geometry of polynomials and a theory of cost. I. *Ann. Sci. École Norm. Sup. (4)*, 18(1):107–142, 1985.
70 M. Shub and S. Smale. Computational complexity: on the geometry of polynomials and a theory of cost. II. *SIAM J. Comput.*, 15(1):145–161, 1986.
71 S. Smale. The fundamental theorem of algebra and complexity theory. *Bull. Amer. Math. Soc. (N.S.)*, 4(1):1–36, 1981.
72 S. Smale. Newton method estimates from data at one point. In R. E. Ewing, K. I. Gross, and C. F. Martin, editors, *In the Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics*, pages 185–196. Springer-Verlag, 1986.
73 A.J. Sommese and C.W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.
74 V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:352–356, 1969.
75 A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.
76 J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Küchlin, editor, *Proc. ISSAC '97*, pages 17–20, Maui, Hawaii, July 1997.
77 J. van der Hoeven. Fast evaluation of holonomic functions. *TCS*, 210:199–215, 1999.
78 J. van der Hoeven. Fast evaluation of holonomic functions near and in singularities. *JSC*, 31:717–743, 2001.
79 J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.
80 J. van der Hoeven. Computations with effective real numbers. *TCS*, 351:52–60, 2006.
81 J. van der Hoeven. Around the numeric-symbolic computation of differential Galois groups. *JSC*, 42:236–264, 2007.
82 J. van der Hoeven. Efficient accelero-summation of holonomic functions. *JSC*, 42(4):389–428, 2007.
83 J. van der Hoeven. New algorithms for relaxed multiplication. *JSC*, 42(8):792–802, 2007.
84 J. van der Hoeven. Fast composition of numeric power series. Technical Report 2008-09, Université Paris-Sud, Orsay, France, 2008.
85 J. van der Hoeven. Making fast multiplication of polynomials numerically stable. Technical Report 2008-02, Université Paris-Sud, Orsay, France, 2008.
86 J. van der Hoeven. Newton's method and FFT trading. *JSC*, 45(8):857–878, 2010.
87 J. van der Hoeven. Efficient root counting for analytic functions on a disk. Technical report, HAL, 2011. `http://hal.archives-ouvertes.fr/hal-00583139/fr/`.
88 J. van der Hoeven. Reliable homotopy continuation. Technical report, HAL, 2011. `http://hal.archives-ouvertes.fr/hal-00589948/fr/`.
89 J. van der Hoeven, G. Lecerf, B. Mourain, et al. Mathemagix, 2002. `http://www.mathemagix.org`.
90 J. Verschelde. *Homotopy continuation methods for solving polynomial systems*. PhD thesis, Katholieke Universiteit Leuven, 1996.
91 J. Verschelde. PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999. Algorithm 795.
92 J.W. von Gudenberg. Interval arithmetic on multimedia architectures. *Reliable Computing*, 8(4), 2002.
93 K. Weihrauch. *Computable analysis*. Springer-Verlag, Berlin/Heidelberg, 2000.